VAASAN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

2010

VAASAN AMMATTIKORKEAKOULU

Alma Stankovic

WEB-PROJEKTIN AIKAINEN TESTAUS

Liiketalous ja matkailu
2010

ALKUSANAT

Tämän opinnäytetyön kirjoittamisen prosessi on ollut pitkä ja haastava. Paljon muutoksia mahtui matkaan, ja välillä motivaatio katosi kokonaan. Haluaisin kiittää kaikkia, jotka ovat minua tässä projektissa tukeneet ja kannustaneet. Erityiskiitokset työkavereille Pirkka Sukselle ja Henna Pynttärille!

VAASAN AMMATTIKORKEAKOULU

Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Alma Stankovic
Opinnäytteen nimi	Web-projektin aikainen testaus
Vuosi	2010
Kieli	suomi
Sivumäärä	45+ 6 liitettä
Ohjaaja	Mika Tamminen

Tässä opinnäytetyössä esitetään kuinka yrityksessä voi kehittää testausta osana isompaa kokonaisuutta, eli projektin prosessia, tässä tapauksessa www-projektia. Tavoitteena on ollut kehittää Visualweb Oy :lle sovellettu testausprosessi, josta tulee omaksuttu ja hyväksi todettu tapa toimia testauksen suhteen projektin aikana.

Kun aloitin työskentelyn testaajana Visualweb Oy :ssä, havaitsin kehittämisen tarpeen. Koko kehittämisprojekti alkoi keräämällä tietoa nykyisestä toimintatavasta kyselyillä. Olen yrittänyt hahmottaa vakavimmat ongelmat ja kehityksen tarpeessa olevat osat testauksessa. Työn olen aloittanut vuonna 2007 tutustumalla testauksen teoriaan ja erilaisiin testausmenetelmiin. Olen käynyt Tieturi Oy :n Testauksen valmennusohjelma - sekä Web-sovellusten testaus - kurssit, joilla käsiteltiin koko testauksen prosessi ja testauksen tavat.

Haasteina tässä työssä ovat olleet suuret järjestelmien vaihdot sekä aikataulun pitäminen. Testaus on periaatteessa pysynyt samankaltaisena, mutta järjestelmien muuttumiset ovat vaatineet myös testauksen menetelmien muuttamista. Testaustavat ja prosessi on kuitenkin saatu yrityksessä toimimaan.

Avainsanat Testaus, prosessi, kehittäminen, web-projekti

VAASA POLYTECHNIC

Tietojenkäsittely

ABSTRACT

Author	Alma Stankovic
Topic	Testing during a web project
Year	2010
Language	Finnish
Pages	45+ 6 appendices
Name of supervisor	Mika Tamminen

This thesis examined how to develop testing as a part of a bigger concept such as a web project process, in a company. The goal was to develop a testing process for Visualweb Oy that would become a good way of testing projects.

When I started to work for Visualweb Oy as a tester I noticed that some things needed improvement. The project started by gathering information about the present situation, the most serious problems and the things that needed improving in testing trough a questionnaire. The work started in 2007 by reading about testing theory and different testing methods. I have also taken two courses at Tieturi: prep course for testing and a Web-software testing course. The courses were about testing methods and the whole testing process in general.

The challenges in this work were major changes taking place in our system and in keeping the schedule. Testing has basically remained the same but system changes have required changes in testing methods. The company has been able to develop well-working testing methods together with a well-working testing process.

Keywords Testing, Process, Developing, Web Project

SISÄLLYS

1. JOHDANTO

Testaus on prosessina erittäin tärkeä osa-alue ohjelmistoprojekteissa. Testausta suoritetaan kuitenkin erittäin vähän tai vajavaisesti. Syy tähän on yleensä se, ettei yrityksessä ymmärretä testauksen tärkeyttä ja sen tuomia hyötyjä. Muita syitä ovat osaamisen puute, yrityksen resurssit ja projektinjohtamisen puute.

Testausprosessi itsessään pitää sisällään monia eri vaiheita, joiden suunnittelemiseen, toteuttamiseen ja hallintaan tarvitaan paljon erilaista osaamista. Testausprosessi sopeutetaan aina projektiin, joten testausta täytyy osata ”säätää” projektin ja tilanteen mukaan. Testausprosessia kehitetään koko ajan yrityksen ja ennen kaikkea järjestelmien kasvun mukaan. Mitä suuremmista ja monimutkaisemmista järjestelmistä on kyse, sitä enemmän vaaditaan selkeää ja systemaattista testausta.

Visualweb Oy on kolmella paikkakunnalla toimiva ohjelmistotalo, päätoimipiste sijaitsee Vaasan Sience Parkissa. Yrityksessä on tällä hetkellä 24 työntekijää, joista 15 Vaasassa. Visualweb Oy on luonut oman sisällönhallintatyökalun www-sivustoille; vuonna 2008 julkaistiin Service-järjestelmä, ja tällä hetkellä työkaluja aktiivisessa käytössä on kaksi kappaletta. Asiakkaina ovat mm. Vaasan kaupunki, Martela, Salcomp ja Kotipizza. Asiakasmäärä kasvaa hurjaa vauhtia, koko ajan tehdään isompia ja vaativampia projekteja, eikä Service venynyt tarpeeksi hyvin vastaamaan tarpeita. Visualweb on vuoden 2009 aikana siirtynyt SharePoint 2007 -ympäristöön, mikä oli yritykselle ja työntekijöille iso haaste. Vuonna 2010 otettiin käyttöön SharePoint 2010.

Tässä työssä on kerrottu yleisellä tasolla testauksesta sekä siitä, miten testaus on edennyt Visualweb Oy :ssa ,ja miten sitä nykyään käytetään. Sharepoint 2010 :n osalta testausta sovelletaan parhaillaan uuteen ympäristöön.

2. MITÄ ON OHJELMISTOTESTAUS?

2.1 Mitä testaus on ja miksi se on tärkeää?

Testaus on vikojen järjestelmällistä etsimistä, laadun varmistamista, todentamista ja kelpuuttamista. Syitä testaamisen on monia. Päällimmäisin syy on sovelluksen toiminnallisuuden ja laadun varmistaminen sekä tuotekehityksen vauhdittaminen. (Tieturin kurssimateriaali 2007)

Testauksen aloittamisen ja verifioinnin aloittaminen mahdollisimman aikaisin säästää paljon vaivaa ja rahaa. Mitä aikaisemmassa vaiheessa virheitä löydetään, sitä helpommin ja halvemmalla ne ovat yleensä korjattavissa. Testaussuunnitelma on tämän takia hyvä tehdä jo heti projektin alussa, samalla kun projektille tehdään määrittelyä. Hyvällä, oikein kohdennetulla testauksella optimoidaan liiketoiminnalle aiheutuvia kustannuksia ja maksimoidaan siitä saatava hyöty. (Tieturin kurssimateriaali 2007)

Kun puhutaan vain webistä, selaintuki on tärkeää, koska eri selaimia on paljon ja niistä on useita eri versioita. Melkein kaikki selaimet käsittelevät koodia hieman eri tavalla, ja huono käyttökokemus webissä luo helposti huonon kuvan yrityksestä, yritys saattaa pahimmillaan menettää mahdollisen asiakkaan ja täten myös rahaa. Nykyään web-sivut ovat kuin yrityksen käyntikortti, ja huono käyttökokemus tai puutteelliset sivustot jättävät huonon kuvan yrityksestä, ja täten yrityksen palveluista tai tuotteista.

Selaintestauksessa on otettava huomioon asiakkaan tarpeet ja vaatimukset, sillä pelkillä uusimmilla selaimilla testaaminen ei ole tarpeeksi. Isoissa organisaatioissa saattaa olla muita ohjelmistoja, jotka jarruttavat uusimpien selainversioiden käyttöönottoa, siksi on tärkeää suorittaa testaus myös vanhemmilla selaimilla, asiakkaan tarpeita silmällä pitäen. Mobiililaitteet ovat koko ajan lisäämässä osuuttaan, joten niiden huomioiminen testauksessa on myöskin tärkeää.

Kun puhutaan intranetistä, selaintestaus ei nouse yhtä tärkeäksi asiaksi, sillä suuremmissa organisaatioissa on yleensä määritetty pelkästään yhden selaintyypin

ja version käyttö. Intraneteissä on tärkeää saada käyttökokemus ja käytettävyys hyväksi, jotta järjestelmä otettaisiin käyttöön nopeasti. Mitä nopeammin intra tai esimerkiksi uusi tilausjärjestelmä otetaan käyttöön, sitä nopeammin se maksaa itsensä takaisin.

Käyttäjien tietokoneosaaminen on eritasoista, jotkut osaavat itse välttää virhetilanteita, jos huomaavat sellaisen mahdollisuuden, ja jotkut eivät virheen ilmetessä edes osaa ajatella mistä virhe olisi voinut syntyä. Järjestelmästä on pyrittävä karsimaan niin sanotut helpot virhetilanteet, jotta ne eivät aiheuttaisi käyttäjälle harmia käytettäessä järjestelmää. Esimerkiksi jos järjestelmässä on kalenteriominaisuutena aloitus- ja päättymispäivämäärä, siinä estetään päättymispäivämäärän olevan pienempi kuin aloituspäivämäärä. Jos kyseessä on lähetettävä lomake, jonka tiedot tallettuvat tietokantaan, estetään liian pitkien syötteiden kirjoittaminen kenttiin, sillä jos kanta vastaanottaa vähemmän syötteitä kuin mitä kenttään saa kirjoittaa, aiheutuu siitä virhetilanne.

Regressiotestaus on erittäin tärkeä osa testauksesta, sillä usein kun sovelluksiin lisätään uusia ominaisuuksia tai muutetaan olemassa olevia ominaisuuksia, niihin kytköksissä olevat osat saattavat vioittua tai lakata toimimasta oikein. On tärkeää tunnistaa ongelmakohdat ajoissa ja suorittaa regressiotestaus kunnolla.

2.2 Testauksen tavoitteet

Testauksen tarkoitus on varmistaa ja parantaa ohjelmiston laatua. Toisaalta testauksen tarkoituksena on yksinkertaisesti ohjelmassa esiintyvien virheiden etsiminen ja korjaaminen. Mitä virheellä sitten tarkoitetaan? Yleisesti ottaen virheeksi voidaan käsittää mikä tahansa tila tai tapahtuma, joka poikkeaa odotetusta. Esimerkiksi ohjelmassa voi tapahtua jotain, mitä sen määrittelyn mukaan ei olisi pitänyt tapahtua. Toisaalta itse määrittely voi olla virheellinen, jolloin virhe tapahtuu, vaikka ohjelma toimisikin määritellysti.

Koska ohjelmaa ei kuitenkaan voida testata täydellisesti, ei koskaan myöskään voida olla täysin varmoja ohjelman virheettömyydestä. Tästä syystä onkin haastavampaa ja järkevämpää asettaa testauksen tavoitteeksi, ohjelman virheettömyyden todistamisen sijaan, mahdollisimman monen virheen löytäminen ja korjaaminen. Käytännössä tämä edellyttää esimerkiksi testiaineiston valintaa siten, että testitapauksilla olisi mahdollisimman suuri todennäköisyys paljastaa mahdolliset virheet. Jotta virheet löydettäisiin mahdollisimman varhaisessa ohjelmistoprojektin vaiheessa, tulisi testausta tehostaa erityisesti matalampien tasojen testauksessa.

Mitä myöhäisemmässä vaiheessa ohjelmistoprojektia virhe löydetään, sitä enemmän työtä sen korjaus teettää, ja sitä kalliimmaksi korjaus tulee. Yleisesti on esitetty, että jos virheen havaitseminen ja korjaaminen ohjelman suunnitteluvaiheessa tulee maksamaan yhden rahayksikön, tulee havaitseminen juuri ennen testausta maksamaan noin 6, testauksen aikana noin 15 ja ohjelman julkaisun jälkeen 60-100 rahayksikköä. Joissakin tapauksissa julkaisun jälkeen havaittu virhe voi itsessään aiheuttaa vielä paljon suurempia kustannuksia. (Ron Patton 2002)

2.3 Testauksen haasteet

Testauksen suurimpana haasteena on testauksen vähättely, eli organisaatiossa ei ymmärretä testauksen tärkeyttä ja sen tuomia hyötyjä projektille. Jos testauksen tärkeyttä ei ymmärretä, on sen arvostus huono. Organisaatiossa esimerkiksi ei ole testauspäällikköä, mutta organisaatiossa on melkein aina projektipäällikkö ja myyntijohtaja. Jos kukaan ei johda testausta, sitä ei usein ajatella ajoissa, ei resurssoida eikä työntekijöitä kouluteta testaukseen tarpeeksi. (Tieturin kurssimateriaali 2007)

Projekti aloitetaan määrittelyllä, mutta testausta ei oteta määrittelypalavereihin mukaan. Tästä seuraa se, että projektilla on usein epärealistinen aikataulu, testausta ei oteta huomioon tai testauksen aika-arvio ei ole realistinen. Projektiin ei tehdä testaussuunnitelmaa, joten testaus vaikeutuu entisestään. Testaukseen sisältyy erittäin paljon manuaalista työtä. Testauksen ajoitus ja aikataulutus on tärkeää, sillä ei ole samantekevää koska esimerkiksi suorituskykyä testataan. Jos testaussuunnitelmaa ei ole tehty, on testaus paljon työläämpi ja aikaavievämpi prosessi. Jos määrittelyä ei ole tehty kunnolla, vaikeuttaa se testaussuunnitelman tekoa, mutta myös koko projekti kärsii. (Tieturin kurssimateriaali 2007)

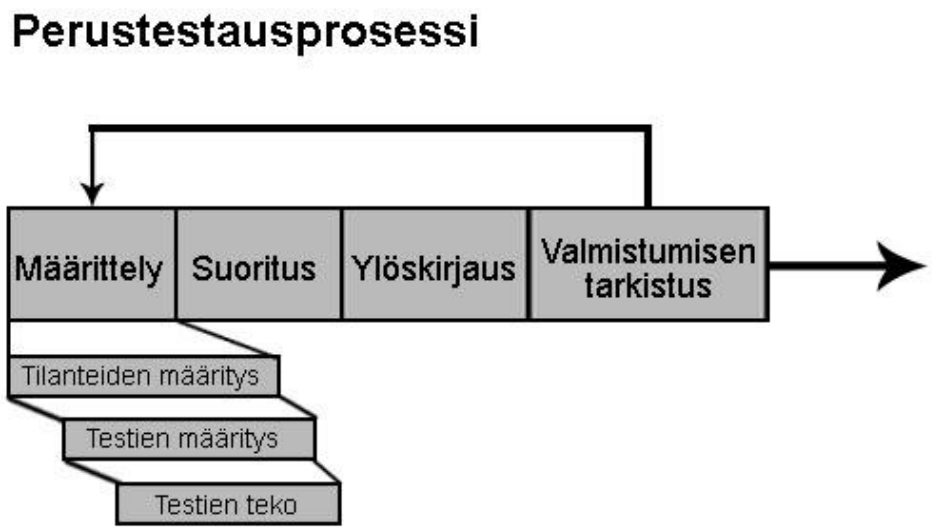
Testaamisen haasteena voi myöskin olla testaajien riittämätön koulutus. Testausta ei osata suunnitella oikein, sitä ei osata hallita eikä toteuttaa oikein. Testaajat eivät välttämättä osaa sopeuttaa testausta projektin vaatimalla tavalla tai projektin tilanteen mukaan. Testaajat eivät osaa kohdentaa testausta oikein ja hukkaavat aikaa, eivätkä testaa projektin tärkeimpiä osia. Testaajien asenne vaikuttaa myös paljon testaustuloksiin, testaajien täytyy olla motivoituneita löytämään virheet. (Tieturin kurssimateriaali 2007)

Koska myös ohjelmoijat suorittavat testausta, voivat ohjelmoijien ja testaajien välit olla ”kireät”. Ohjelmoijilla on usein väärä asenne; ”Kyllä sen pitäisi toimia, vaikka en paljon testannut!” Ohjelmoijat eivät usko, että heidän koodissaan voi olla suurempia virheitä, ja kun niitä ilmenee, ei olla tyytyväisiä. Ohjelmoijien ja testaajien tulisi pitää kiinni ja ylläpitää yhdessä olemassa olevia laatustandardeja. (Tieturin kurssimateriaali 2007)

3. MITÄ ON TESTAUSPROSESSI?

3.1 Testausprosessi pähkinänkuoressa

Testausprosessi, kuten mikä tahansa prosessi, alkaa ja loppuu johonkin. ISEB:n (Information Systems Examinations Board) määrittelemään perustestausprosessiin kuuluvat (Kaavio 1):



Kaavio 1: Perustestausprosessi (ISEB - Information Systems Examinations Board). (Tieturi kurssimateriaali 2007)

Määrittelyn ja suorituksen vaiheisiin kuuluvat tilanteiden määritys, testien määritys ja testien teko. Tässä vaiheessa päätetään mitä testataan ja priorisoidaan tärkeimmät testitapaukset. Päätetään kuinka testataan ja asetetaan aikataulu. (Tieturin kurssimateriaali 2007)

Kaavio selitettynä sanallisesti: Määrittelyvaiheessa määritellään tilanteet, eli järjestelmän eri osa-alueet. Tämän jälkeen määritellään, eli suunnitellaan erilaisia testejä, joita tullaan suorittamaan. Suunnittelemisen jälkeen siirrytään varsinaiseen suoritukseen, eli varsinaisten testien tekoon määritelmien mukaan.

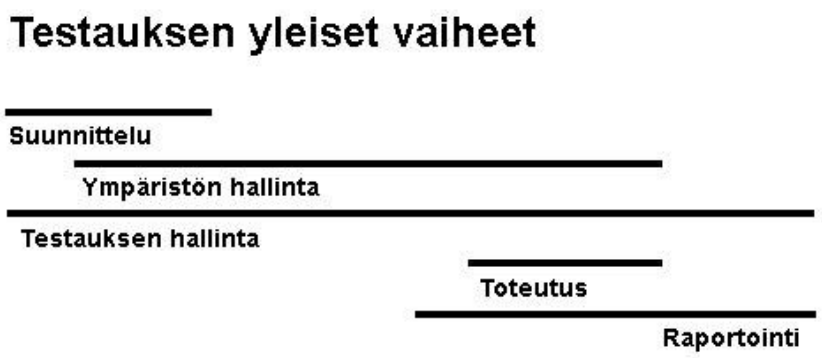
Omassa yrityksessä pitää osata arvioida, määritellä ja päättää mistä testausprosessi alkaa ja mihin se päättyy.

3.2 Testausprosessi osana projektiprosessia

Testausprosessi on pieni, mutta tärkeä osa projektin prosessia. Testauksessa on monta eri osa-aluetta ja erityyppistä testausmenetelmää.

Testauksen yleisiin vaiheisiin kuuluvat (Kuvio 1):

- suunnittelu
- ympäristön hallinta
- testauksen hallinta
- testaus ja raportointi



Kuvio 1: Testauksen yleiset vaiheet.

Suunnittelu, ympäristön hallinta ja toteutuksen hallinta kuuluvat tai perustuvat määrittelyvaiheeseen. Suunnitteluvaiheessa määritellään kuinka hallitaan ympäristöä ja testausta, eli suunnitellaan testausprosessin kulku. (Tieturin kurssimateriaali 2007)

Testauksen hallinta kestää koko tämän prosessin ajan. Siihen taas kuuluvat kaikki määrittelystä raportointiin ja valmistumisen tarkistamiseen.

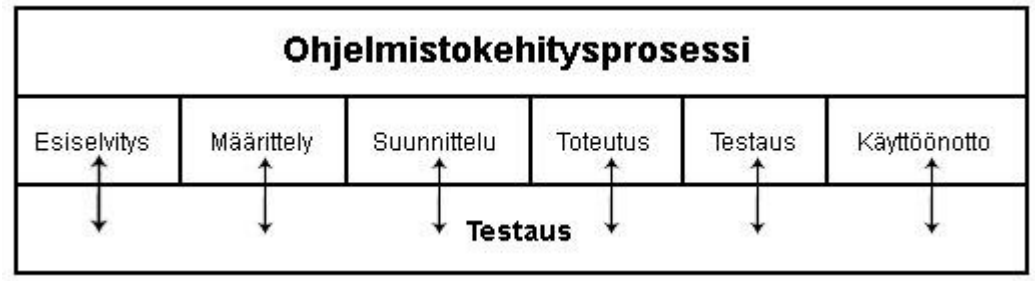
Toteutus on itse tekemistä, eli suunniteltujen testien suorittaminen. Toteutus kuuluu suoritusvaiheeseen.

Raportointivaiheessa tehdään testiraporttipohjat, testiraportit täytetään ja virheraportit luovutetaan ohjelmoijalle. Raportointi kuuluu suoritus-, ylöskirjaus- ja valmistumisen tarkistusvaiheeseen.

3.3 Testaus osana ohjelmistokehitysprosessia

Testaus jota tehdään ohjelmistokehitysprosessin (Kuvio 2) aikana, voidaan jakaa vaiheisiin:

- määrittelyn ja suunnittelun aikainen testaus
- kehityksen aikainen testaus
- ylläpidon aikainen testaus



Kuvio 2: Ohjelmistokehitysprosessi.

Testausta tehdään koko ohjelmiston elinkaaren ajan. Käytön ja ylläpidon aikainen testaus muodostaa pääosan ohjelmiston kokonaiskustannuksista (keskiarvo 67%).

Määrittelyn ja suunnittelun aikainen testaus suoritetaan staattisena ja dynaamisena testauksena. Staattiseen testaukseen kuuluu vaatimusmäärittelyn, toiminnallisen määrittelyn, teknisen määrittelyn, teknisen suunnittelun ja testaussuunnittelun dokumenttien testaus katselmoinnin avulla. Dynaamiseen testaukseen kuuluu mallien simulointi, suorituskyvyn analysointi ja prototyyppien testaus. (Tieturin kurssimateriaali 2007)

Hyvin suunniteltu ja läpiviety testausprosessi parantaa laatua, nopeuttaa projektia ja säästää kustannuksia. (Tieturin kurssimateriaali 2007)

4. TESTAUSTASOT JA –MENETELMÄT

Testausmenetelmillä tarkoitetaan tapaa suorittaa testit. Jokainen testausmenetelmä voidaan liittää johonkin testaustasoon. Testitapausten määrittely tapahtuu valitsemalla ensin testimenetelmä. Yleisimmin käytetään lasilaatikkotestausta, mustalaatikkotestausta tai harmaalaatikkotestausta.

4.1 Moduulitestaus eli yksikkötestaus

Moduulitestaus on järjestelmän pienin testattavissa oleva osa. Moduuli on kooltaan tyypillisesti alle 1000 koodiriviä ja yleensä yhden ohjelmoijan tekemä. Moduuleja eli yksiköitä testataan, jotta voidaan osoittaa yksikön joko olevan vaatimusten mukainen tai korjauksen vaativa. Virheitä kutsutaan yksikkövirheiksi. Koska moduulitestauksessa keskitytään vain pienempien osien testaukseen, ei se sovellu ainoaksi testausmenetelmäksi. Yleisin menetelmä on lasilaatikkotestaus. (Tieturin kurssimateriaali 2007)

4.1.1 Lasilaatikkotestaus eli White box testing

Lasilaatikkotestauksessa hyödynnetään tietoa ohjelman toteutuksesta testitapausten valinnassa. Tavoitteena on, että kaikki ohjelman haarat ja ohjelmapolut saadaan käytyä läpi. Moduulit pystytään tällöin käymään läpi erittäin tarkalla tasolla, ja testaaja näkee mitä on jo käyty läpi. Tätä menetelmää käytetään lisäämään loogista kattavutta, jota on neljää peruslajia:

- Lausekkeet
- Valinnat ja haarat
- Ehdot
- Polut

(Tieturin kurssimateriaali 2007)

4.2 Integraatiotestaus

Integraatiotestauksessa moduuleja yhdistetään suuremmiksi kokonaisuuksiksi ja rajapintoja testataan. Rajapintojen testaus onkin tässä tärkein testauksen kohde. Integraatiotestausta suoritetaan usein yksikkötestauksen rinnalla, ja siinä varmistetaan, että moduulit toimivat oikein yhdessä. Testaus voidaan suorittaa lisäämällä moduuleja kokonaisuuteen yksi kerrallaan, eli niin sanotulla lisäävällä menetelmällä (incremental) tai sitten yhdistämällä kaikki moduulit heti, eli niin sanottulla ei-lisäävällä menetelmällä (non-incremental). Lisäävässä menetelmässä testataan ensin yksi lisätty moduuli järjestelmässä, ja kun se on todettu toimivaksi, lisätään yksi moduuli lisää. Ei-lisäävässä menetelmässä yhdistetään kaikki moduulit ja testataan koko järjestelmä. Tämä on kuitenkin haastavampi ja heikompi menetelmä, sillä virheiden jäljittäminen on paljon vaikeampaa. Yleisin menetelmä on mustalaatikkotestaus. (Tieturin kurssimateriaali 2007)

4.2.1 Mustalaatikkotestaus eli Black box testing

Mustalaatikkotestauksessa testitapaukset valitaan usein määrittelyiden perusteella. Menetelmä on tehokas suuremmille koodimäärille, ja testitapauksia tehdään myös käyttäjän näkökulmasta. Tässä menetelmässä testaaja ja ohjelmoija eivät ole riippuvaisia toisistaan, vaan testaaja voi toimia itsenäisesti. Testitapausten valintamenetelmät ovat ekvivalenssiosoitus, arvotestaus, syy-seurausanalyysi ja virheenarvaus. Tätä testausmenetelmää käytetään integraatiotestauksen lisäksi myös järjestelmä- ja hyväksymistestauksessa. (Tieturin kurssimateriaali 2007)

4.3 Systeemitestaus eli järjestelmätestaus

Systeemitestauksessa testataan järjestelmää kokonaisuudessaan sen käyttötarkoitusta vastaavassa ympäristössä. Järjestelmää testataan loppukäyttäjän näkökulmasta ja pyritään varmistamaan, että koko järjestelmä toimii määrittelyjen ja vaatimusten mukaisesti. Systeemitestaus on yksi tärkeimmistä osista, kun testausta ajatellaan laajasti, se kuluttaa suurimman osan testauksen resursseista. Systeemitestaukseen kuulu myös ei-toiminnallinen testaus. Yleisin menetelmä on harmaalaatikkotestaus. (Tieturin kurssimateriaali 2007)

4.3.1 Harmaalaatikkotestaus

Harmaalaatikkotestaus on lasi- ja mustalaatikkotestauksen sekoitus, siinä on pyritty yhdistämään kummankin menetelmän hyviä puolia. Mustalaatikkotestaus saattaa jäädä vajavaiseksi ja lasilaatikkotestaus voi olla liian yksitoikkoinen, joten näiden yhdistelmä tuottaa parempia tuloksia. Testitapaukset täytyy valita tarpeen mukaan, jolloin ei jouduta siihen tilanteeseen, että testausmallin määrittelyt rajoittaisivat testausta. Testausta suunnitellaan ympäristön ja teknisen toteutuksen tuntemiseen perustuen. Harmaalaatikkotestaus on tärkeä testaustapa esimerkiksi web-sovelluksissa. (Tieturin kurssimateriaali 2007)

4.4 Hyväksymistestaus

Hyväksymistestauksen suorittavat asiakkaat ja/tai käyttäjät, ja itse testaaja voi olla testauksessa mukana. Tällä menetelmällä varmistetaan, että ohjelmisto täyttää tilaajan asettamat vaatimukset. Ajallisesti tämä testaus suoritetaan aivan viimeisimpänä vaiheena. Hyväksymistestauksella lisätään käyttäjän luotettavuutta järjestelmään. (Tieturin kurssimateriaali 2007)

4.5 Regressiotestaus

Virheiden korjausten jälkeen testataan uudestaan – varmistetaan virheet korjatuiksi, ja että korjaukset eivät ole aiheuttaneet uusia virheitä. Usein regressiotestaus suoritetaan kiireellä ja vajavaisesti. Regressiotestauksen laiminlyönnin seurauksena on isompia ja työläämpiä ongelmia. (Tieturin kurssimateriaali 2007)

4.6 Ei-toiminnallinen testaus eli Non Functional Testing

Ei-toiminnallisella menetelmällä testataan suorituskykyä, turvallisuutta ja käyttöoikeuksia. Tämä menetelmä kohdistuu järjestelmän ominaisuuksiin, jotka vaikuttavat ohjelmistoon liittyvään kokonaislaatukokemukseen, mutta eivät ole liitettävissä suoraan toimintoon tai toimintoryhmään.

Suorituskyvyn testauksessa sivusto kuormitetaan, ja katsotaan kuinka sivusto vastaa pyyntöihin ja kuinka palvelin kestää kuormitusta. Tällä tavalla nähdään, ovatko vasteajat halutuilla käyttäjämäärillä siedettäviä (liite 1).

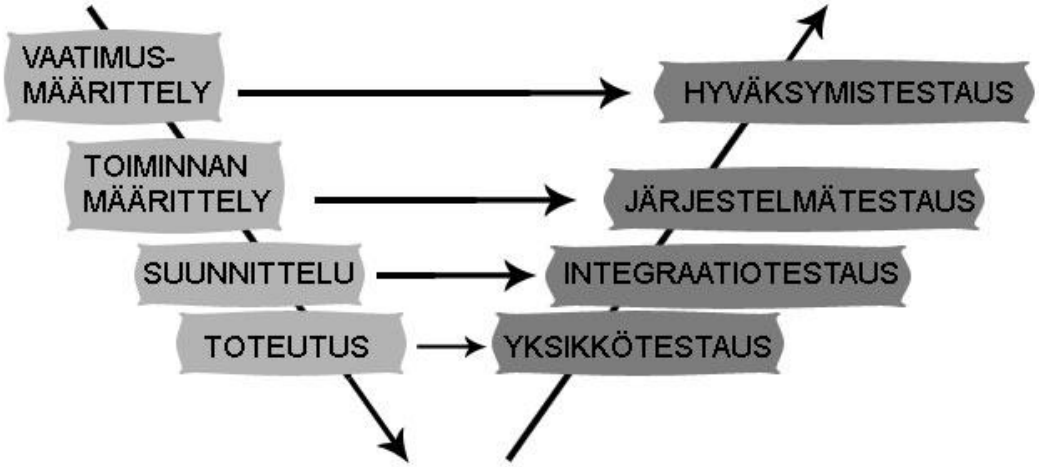
Turvallisuuden testaus (Security testing) on vaativa alue, johon tarvitaan suurempaa osaamista. Perusasiat kuitenkin pystytään testaamaan ilman suurempaa osaamista, voidaan esimerkiksi etsiä kohtia, joissa turvallisuus ei toimi kuten pitäisi. Annetaan oikeita ja vääriä tunnuksia ja salasanoja sekä yritetään tunkeutua järjestelmään, lisäksi voidaan myös tarkastella toimivatko määritetyt käyttöoikeudet oikealla tavalla, eli esimerkiksi, että käyttäjä suppeammilla oikeuksilla ei näe järjestelmästä osia, jotka on tarkoitettu pääkäyttäjille. (Tieturin kurssimateriaali 2007)

5. VAIHEJAKOMALLIT

Vaihejakomallilla tarkoitetaan tapaa, jolla ohjelmiston kehitystyö tai koko elinkaari jaetaan vaiheisiin (Haikala & Märijärvi 2001). Haikalan ja Märijärven (2001) mukaan ”mallista on olemassa useita eri muunnelmia, mutta yleensä niistä voidaan erottaa ainakin määrittely-, suunnittelu- ja toteutusvaiheet. Määrittelyvaihetta edeltää usein esitutkimukseksi (feasibility study, preliminary analysis) tai tarvekartoitukseksi (requirements study) kutsuttu vaihe.” Vesiputousmallissa toimintaohje on Haikalan ja Märijärven (2001) mukaan: ”Analysoi ratkaistava ongelma niin, että ymmärrät sen kunnollisesti, suunnittele ratkaisu, toteuta se ja testaa ratkaisua.” Kirjoittaja huomauttavat, että ”käytännön ohjelmistokehitys ei voi koskaan edetä kirjaimellisesti vesiputousmallin mukaisesti, mm. koska osa vaatimuksista selviää vasta projektin aikana”.

5.1 V-malli

V-malli (Kuvio 3) on testauksen näkökulmasta katsottuna yksi käytetyimmistä malleista. V-malli on edelleen kehitetty vesiputousmallista ottaen huomioon testauksen osuus jokaisessa projektin vaiheessa. Projekti etenee V-mallin mukaan vasemmalta asteittain tarkentuen seuraavassa järjestyksessä: vaatimusmäärittely, toiminnan määrittely, suunnittelu ja toteutus. Oikealta alhaalta alkaa sitten eri tasoilla suoritettava testaus, joka nojautuu eri vaiheissa tehtyihin dokumentteihin. (Tieturin kurssimateriaali 2007)

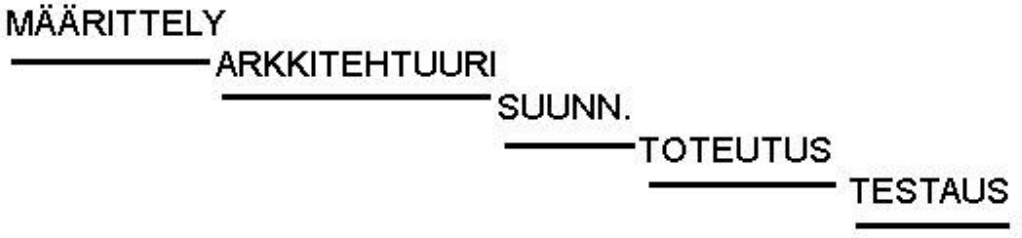


Kuvio 3: V-malli.

Eri testausvaiheiden suunnittelu voidaan aloittaa, kun sitä vastaava vaatimus- ja suunnitteluvaihe on valmis. Testaus alkaa yksikkötestauksella, jossa keskitytään yksittäisten osien testaukseen, ja testaus loppuu hyväksymistestaukseen, jossa asiakas ja testaaja yhdessä toteavat järjestelmän vastaavan vaatimusmäärittelyä ja toimivan oikein.

5.2 Vesiputousmalli

Vesiputousmalli (Kuvio 4) on perinteinen tapa, joka sopii projekteihin, joissa voidaan tehdä tarkat määrittelyt. Malli sopii pienempiin projekteihin, joissa määrittely ei tule muuttumaan myöhemmässä vaiheessa projektia. Sopii esimerkiksi kun vanha, tarkkaan määritelty projekti siirretään uuteen ympäristöön. (Tieturin kurssimateriaali 2007)

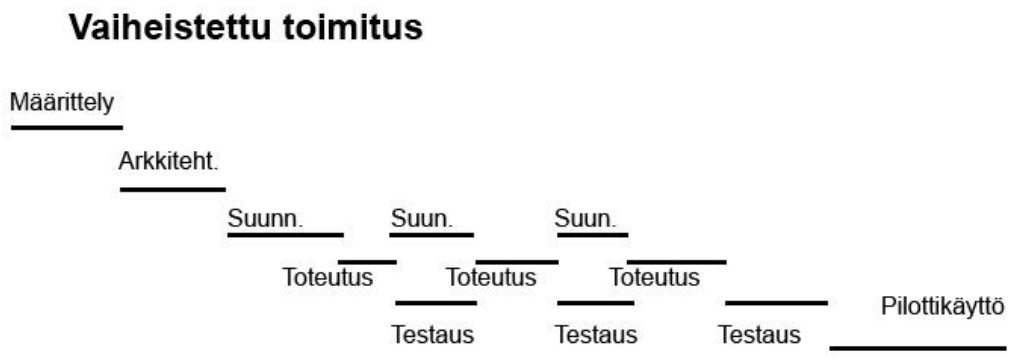


Kuvio 4: Vesiputousmalli.

5.3 Vaiheistettu toimitus

Yleisin tapa toteuttaa projekti on vaiheistettu toimitus (Kuvio 5). Vaatimusmäärittely, toiminnallinen määrittely ja arkkitehtuuri tehdään heti alussa. Kun edellä mainitut on tehty, aloitetaan toteutus järjestelmän osa kerrallaan, ja kussakin vaiheessa suoritetaan tekninen suunnittelu ja toteutus. Tämä malli ei kuitenkaan sovi pienempien web-projektien toteuttamiseen, vaan enemmänkin suuriin projekteihin, joissa järjestelmä toteutetaan vaiheissa. (Tieturin kurssimateriaali 2007)

Vaihestettu malli on kätevä siinä mielessä, että siinä päästään testaamaan ja korjaamaan kunkin vaiheen lopussa, ja viimeisen osajärjestelmän jälkeen ei jää yhtä paljon testattavaa kuin esimerkiksi vesiputousmallissa. Tämä malli vaatii kuitenkin enemmän testausaikaa ja on pitkäaikainen. Viimeisessä osajärjestelmässä saattaa olla huomattavia virheitä, joiden korjaus voi olla mahdotonta ennen järjestelmän suunniteltua käyttöönottoa. (Tieturin kurssimateriaali 2007)

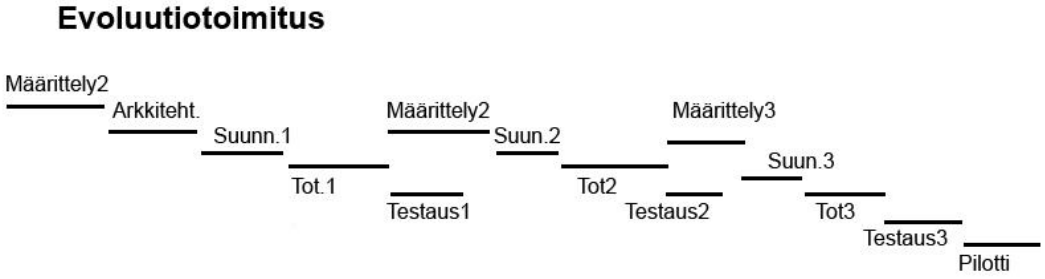


Kuvio 5: Vaiheistettu toimitus.

5.4 Evoluutiotoimitus

Evoluutiotoimituksen (Kuvio 6) ideana on, että kaikkea ei yritetä määritellä tarkasti heti, vaan määritellään ensin arkkitehtuuri ja tärkeimmät tai hankalimmat kohdat järjestelmästä, ja tarkennetaan sisältöä iteroimalla. Tälläinen malli sopii projekteihin, joissa sovellusalue on uusi ja järjestelmä on vaikea määritellä

kerralla kuntoon. Evoluutiotoimitus on hyvä testauksen kannalta, koska osajärjestelmiä päästään testaamaan ja korjaamaan kunkin vaiheen lopussa, näin testattavaa ei jää paljon viimeisen osajärjestelmän jälkeen. Suuret riskit saadaan testattua heti alkuvaiheessa. Tällä mallilla testaus voi kuitenkin mennä hukkaan, jos järjestelmä muuttuu radikaalisti. Testaus on pitkäaikainen ja testauksen tarpeen arviointi on erittäin hankalaa. (Tieturin kurssimateriaali 2007)



Kuvio 6: Evoluutitoimitus.

5.5 Ketterä malli (Agile)

Ketterän kehityksen maailmassa lähdetään siitä, että ei ole olemassa yhtä oikeaa tapaa saavuttaa haluttu lopputulos. Senpä vuoksi harvat ketterät menetelmät ovat tarkoin määriteltyjä ohjeistoja siitä, miten missäkin tilanteessa pitää toimia. Pakollisten käytäntöjen sijaan manifestin taustalla on kokoelma periaatteita, joita noudattamalla uskotaan päästävän hyvään lopputulokseen. (Ketterä kehitys – www-sivu). Tässä mallissa muutokset kesken projektia otetaan vastaan hyvin, ja asiakkaan kanssa kommunikoidaan paljon projektin aikana. Luottamus tekijöiden ja asiakkaan välillä on tärkein osa, tyytyväinen asiakas ja toimiva sovellus ovat pääasia. (Tieturin kurssimateriaali 2007)

6. VIRHERAPORTIT

6.1 Virheraporttien teko

Testaajat tulisi kouluttaa hyvin tekemään virheraportteja, sillä ne ovata tärkeimpiä testaajan työstä syntyviä dokumentteja. Hyvän virheraportin perusteella on helppo päättää kyseisen virhen korjauksen tärkeys sekä löytää ja korjata virhe. Raporttien toteutustavasta on sovittava, jotta se olisi yhtenäinen ja selkeä. Koulutus ja virheraporttien määrittely maksaa itsensä takaisin säästettynä aikana.

6.2 Virheraportin ominaisuudet

Virheraportin tulisi sisältää vain yhden virheen, monta virhettä raportilla vaikeuttaa sen käsittelyä. Yksiselitteinen tunniste virheraporteissa helpottaa luettavuutta, juokseva numerointi riittää yleensä hyvin. Virheraportissa tulee ilmetä miten toimenpiteet ennen virheen ilmenemistä, saattaa olla tarpeellista kertoa myös mitä tehtiin ennen varsinaista toimenpidettä. Liitteet, etenki kuvakaappaukset virheistä auttavat paljon. Virheilmoituksesta tulee ilmetä myös testitapaus, jota suoritettiin, versio, jota testattiin sekä ympäristö, jossa testattiin. Virheraportin virhe tulee olla toistettavissa, jos sitä ei voida toistaa, tulee miettiä, oliko virhe testaajassa tai ympäristössä.

Tulee testata, ilmeneekö virhe muilla arvoilla ja mitkä asiat vaikuttavat virheen syntymiseen, tätä kutsutaan eristäväksi menetelmäksi. Voidaan myös katsoa, onko virhe yleistettävissä, eli ilmeneekö esimerkiksi useammalla syötteellä. Eristys ja yleistys lisäävät testaajan, mutta säästävät ohjelmoijien aikaa. Tulee miettiä, kumman on järkevämpi tämä osuus suorittaa. (Tieturin kurssimateriaali 2007)

7. TESTAUKSEN KEHITTÄMISEN ALOITUS

7.1 Nykytilanteen kartoitus

Kun aloitin työskentelyn testaajana Visualweb Oy :ssä vuonna 2007, jouduin alusta asti kehittämään testausta itse. Yrityksessä ei minua ennen ollut varsinaista testaajaa, eikä valmiita toimintatapoja testauksen suhteen. Testauksen tekeminen oli haasteellista, sillä minulla ei ollut varsinaista koulutusta tai osaamista siihen. Kuitenkin työtä tehdessä havaitsin monia kehityksen tarpeessa olevia seikkoja, josta sain idean tähän opinnäytetyöhön.

Suoritin yrityksessä kyselyjä, joilla yritin hahmottaa työntekijöiden näkemyksen testauksesta ja sen tärkeydestä. Visualweb Oy :ssä työskentelee myyjiä, projektipäälliköitä sekä ohjelmoijia. Kyselyistä kävi selkeästi ilmi, että myyjät eivät ymmärtäneet, mitä testaus oikeastaan pitää sisällään ja kuinka tärkeää se on. Pääasiassa oli ajateltu, että testaus on pelkästään käyttöliittymätestausta eri selaimilla, mutta oli kokonaan unohdettu moduulitestaus sekä integraatiotestaus. Projetipäälliköillä oli samankaltaisia ajatuksia kuin myyjillä, mutta he ymmärsivät enemmän testauksen tärkeydestä projekteissa. Ohjelmoijat puolestaan kertoivat, että pitävät testausta tärkeänä, mutta harmittelivat sen vähäisyyttä.

Kyselyssä kysyttiin mielipiteitä kolmesta tärkeimmästä asiasta testauksessa ja testauksesta syntyvistä hyödyistä. Tärkeimmäksi asiaksi nousi yhdenmukaisuus, eli että testataan samalla tavalla. Seuraava asia oli järjestelmällisyys; testaaja on hyvin perehtynyt sivustoon ja määrittelyyn ja osaa kohdentaa testauksen heikoimpiin kohtiin ja löytää suurimman osan virheistä, jotta projekti nopeutuu. Kolmantena asiana oli laatu, eli tyytyväinen asiakas. Ollaan testattu käyttäjän näkökulmasta ja asiakas saa haluamansa sivuston toimivana.

Visualweb Oy :ssä testausta ei oikein osattu niin sanotusti arvostaa, eikä suurin osa työntekijöistä tiennyt kuinka tärkeä osa projektia testaus on. Testausta ei mitenkään otettu mukaan projektiprosessiin, siihen ei varattu aikaa eikä regressiotestausta suoritettu usein.

Varsinaista projektiprosessia, saatikka testausprosessia, ei yrityksessä ollut. Määrittelyt olivat vajavaisia, ja joskus niitä ei tehty lainkaan, web-sivustot koodattiin ulkoasukuvien perusteella. Pahimmassa tapauksessa testaaja sai eteensä valmiin sivuston ilman määrittelyjä ja tarkempaa tietoa toiminnallisuuksista. Testaus suoritettiin viime tinkaan, eikä aikaa regressiotestaukseen jäänyt usein ollenkaan. Joskus kun aikataulu petti, sivustot julkaistiin ensin ja testattiin myöhemmin. Hyväksymistestausta ei suoritettu, joten testaaja oli erittäin harvoin asiakkaan kanssa tekemisissä.

Yllä mainittujen ongelmien takia tominnallisuuksien ja määrittelyjen selvitys vei paljon aikaa monelta eri ihmiseltä, ja aikataulu venyi tai konkreettista testausaikaa jäi erittäin vähän. Määrittelyn vajavaisuudesta tai puuttumisen johdosta testitapauksia ei voitu suunnitella oikein, joten sivustoihin saattoi jäädä enemmän virheitä. Hyväksymistestauksen puuttumisen johdosta mahdollisten virheiden löytyminen, ja näiden korjaus myöhemmässä vaiheessa, maksoi yritykselle enemmän, sillä niistä ei voitu asiakasta laskuttaa, koska määrittely oli joko vajavainen tai projektia ei niin sanotusti ikinä saatu päätökseen, eli hyväksytyksi.

7.2 Kehityksen tarpeessa olevien alueiden ja tavoitteiden määrittäminen

Testausta prosessina ei otettu huomioon myynnistä alkaen oikein missään. Ohjelmoijat suorittivat jollakin tapaa yksikkötestausta, mutta se ei yksin riitä. Toinen kehityksen tarpeessa oleva asia on projektipäällikköiden suhtautuminen testaukseen. Halusin että testaus otetaan huomioon heti projektin alkuvaiheessa ja aikatauluissa. Testaajan olisi hyvä olla mukana heti määrittelystä alkaen arvioimassa testauksen vaatima aika ja antamassa mielipiteitään projektista ja määrittelystä. Projektin muuttumista tai aikataulujen venymisiä ei otettu useinkaan huomioon. Testaus joko suoritettiin suppeana tai ei ollenkaan, mikä oli erittäin huono asia sekä yrityksen että asiakkaan kannalta. Ohjelmoijat eivät suhtautuneet testaukseen hyvällä asenteella, testaus jäi usein tekemättä ja

virheiden löytäminen jäi kokonaan testaajan tai asiakkaiden tehtäväksi. Hyväksymistestausta ei suoritettu, jolloin asiakas ei varsinaisesti koskaan hyväksynyt projektia. Tästä seurasi se, että ei osattu vetää rajaa korjauksen ja lisätyön välille.

Tavoitteina testauksen suhteen oli saada testaus osaksi projektiprosessia ja tehdä siitä toimiva, kätevä ja omaksuttu tapa Visualweb Oy :ssä. Jotta tämä tavoite onnistuisi, oli testaus saatava mukaan projektiprosessiin ihan myynnistä alkaen. Myynnin oli otettava testaus huomioon siten, että siitä keskustellaan asiakkaan kanssa, ja testausta myydään osana projektia. Jos testaus on selitetty asiakkaalle hyvin, ovat he valmiimpia maksamaankin siitä ja suorittamaan testausta myös itse. Kun testaus on nyt osana projektia tässä vaiheessa, seuraava vaihe on saada projektipäälliköt ottamaan testaaja mukaan määrittelypalaveriin, jotta saadaan myös testaajan näkökulma määrittelyyn ja aikataulutukseen. Näin testaaja on tietoinen projektin osista alusta alkaen, osaa suunnitella testauksen paremmin ja osaa varautua muutoksiin paremmin. Hyväksymistestaus oli otettava käyttöön, jotta mahdollisilta lisäkustannuksilta ja riitelyiltä asiakkaan kanssa vältyttäisiin. Kun asiakas itse testaa ja hyväksyy projektin, ovat mahdolliset lisätyöt aina laskutettavia.

7.3 Koulutus

Testaukseen tutustumiseen hyvä kurssi oli Tieturin Testauksen valmennusohjelma, joka koostui kahden päivän luennosta sekä puolen päivän tentistä, josta sai sertifikaatin (liite 2) kurssilla läpi käydystä testauksen teoriasta. Kurssi käsitteli seuraavia asioita:

- Testauksen peruskäsitteet ja periaatteet
- Testausorganisaation määrittely
    - Testaajan ja testausryhmän roolit ja taidot
- Testausprosessi
    - Testausprosessin kehittämismallit
- V-malli ja testaustasot
    - Yksikkötestaus

    - Integraatiotestaus
    - Systeemitestaus
    - Hyväksymistestaus
    - Regressiotestaus
    - Aloitustesti
- Testauksen ajoitus ja riskiohjattu testaus
    - Testityömäärän arviointi
- Testaussuunnittelu
    - Testaussuunnitelma
    - Testitapaukset
- Testityypit
    - Suorituskyvyn testaus
    - Käytettävyyden testaus
    - Tietoturvan testaus
    - Koodin katselmointi ja laatumetriikat
- Testauksen automatisointi
    - Testien nauhoitus
    - Testiympäristön standardointi
- Testitapauksen perusteet
    - Hyvä testitapaus
    - Testitapauksen laajuus
    - Tarkistuslistat
- Ohjelmistovirheet
    - Hallinta ja raportointi
- Testaustekniikat
    - Raja-arvojen testaus
    - Ekvivalenssiluokat
    - Positiivinen ja negatiivinen testaus
    - Käyttötapauksiin pohjautuva testaus
    - Funktio- ja datalähtöiset tekniikat
    - Tutkiva testaus
- Toiminnallisen testauksen testitapaukset

    - Käyttöliittymätestaus
    - Käyttötapaukset
    - Järjestelmien välisten liittymien testaus
- Testitapauksen kuvaaminen
    - Eritasoiset kuvaukset
- Testauksen raportointi
    - Virheiden hallinnan käytännöt
- Testaustyökalut
    - Työvälineiden jaottelu

Tästä kurssista sai erittäin hyvän pohjan testaukseen, testausprosessiin ja erilaisiin testausmenetelmiin. Testauksen tärkeys tuli esille hyvin, ja kurssilla sai ideoita testauksen ja prosessin kehittämiseen. Sertifikaatti kurssista liitteessä X. Seuraavaksi valitsin Web-sovellusten testaus -kurssin (liite 3), jolla keskityttiin sitten enemmän web-sovellusten testaukseen. Kurssi koostui kahdesta luentopäivästä ja käsitteli seuraavia asioita:

- Miten web-sovelluksen testaus eroaa muusta testauksesta?
    - Erot ja samankaltaisuudet
- Web-järjestelmän kartoitus
    - Testausta edeltävät työt
- Testaus selaimen kautta
    - 22 kohdetta testaukseen
    - Skriptaus, keksit, istunnot
    - Navigoinnin testaus
    - Linkit, hakukoneet
- Tietoturvatestaus
    - Owasp-ohjeistus
    - Haavoittuvuuksien testaus
- Suorituskykytestaus
    - Erilaisten suorituskykytestien suoritus
- Yhteensopivuustestaus

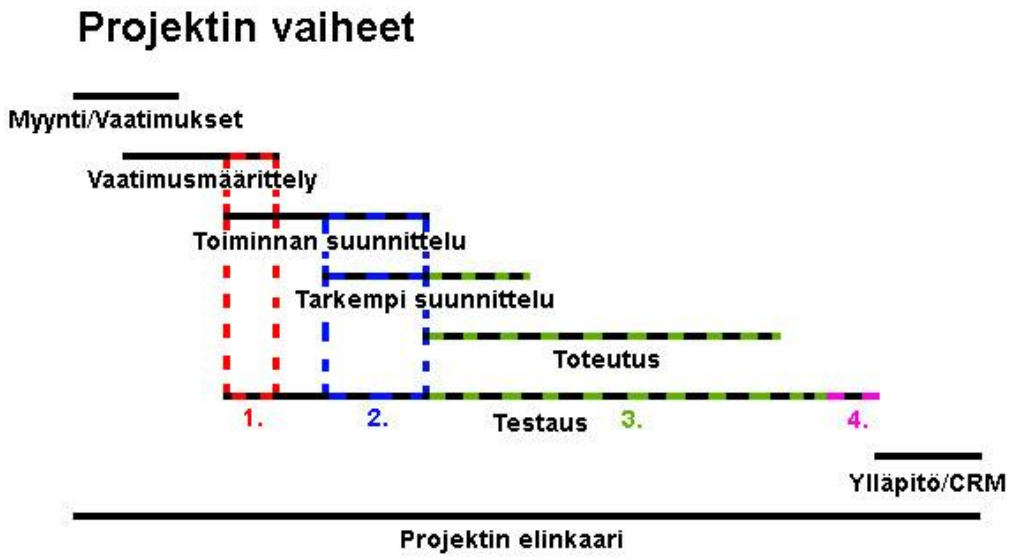
    - Mitä pitää ottaa huomioon web-käyttöliittymätestauksessa
- Web-testauksen työkalut
    - Käyttöliittymätestauksen automatisointi

Tällä kurssilla sai enemmän tietoa testauksesta web-ympäristössä ja mitä tällainen testaus vaatii. Kurssista sai todistusksen osallistumisesta (liite 3).

Testauksesta on olemassa aika vähän koulutuksia, mutta kansainvälisesti arvostetut tutkinnot ovat ISEB:n (Information Systems Examinations Board) järjestämät ISEB Foundation Certificate, ISEB Intermediate Certificate ja ISEB Practitioner Certificate. Kursseista on olemassa valmennuskurssit, kaikki materiaalit ja kokeet ovat englanniksi. Kurssien suorittamisesta saa kansainvälisen ISEB-sertifikaatin. Tavoitteenani on vuoden 2011 loppuun mennessä saada ainakin yksi sertifikaatti.

8. TESTAUS VISUALWEB OY :SSÄ TÄLLÄ HETKELLÄ

Visualweb on palkannut myös toisen testaajan, testauspäällikköä ei kuitenkaan ole, vaan testaajat toimivat itsenäisesti, mutta kuitenkin toisiaan tukien. Visualweb Oy :ssä testaus on tällä hetkellä otettu huomioon alusta asti. Testausta myydään osana projektia, ja suuremmat asiakkaat jopa vaativat testauksen suorittamista sekä hyväksymistestausta. Yrityksen työntekijät on jaettu tiimeihin, jotka koostuvat kahdesta projektipäälliköstä sekä kahdesta ohjelmoijasta, tuotekehitys ja testaus ovat erikseen. Tiimit ottavat testaajat mukaan projekteihin vaatimusmäärittelystä asti (Kaavio 2). Testauksen aikataulutuksessa otetaan huomioon projektin vaiheet ja haasteellisuus. Projekteissa käytetään useimmin ketterää vaihejakomallia, sillä se on Visualweb Oy :ssä parhaaksi todettu malli web-projekteille.



Kaavio 2: Projektin vaiheet.

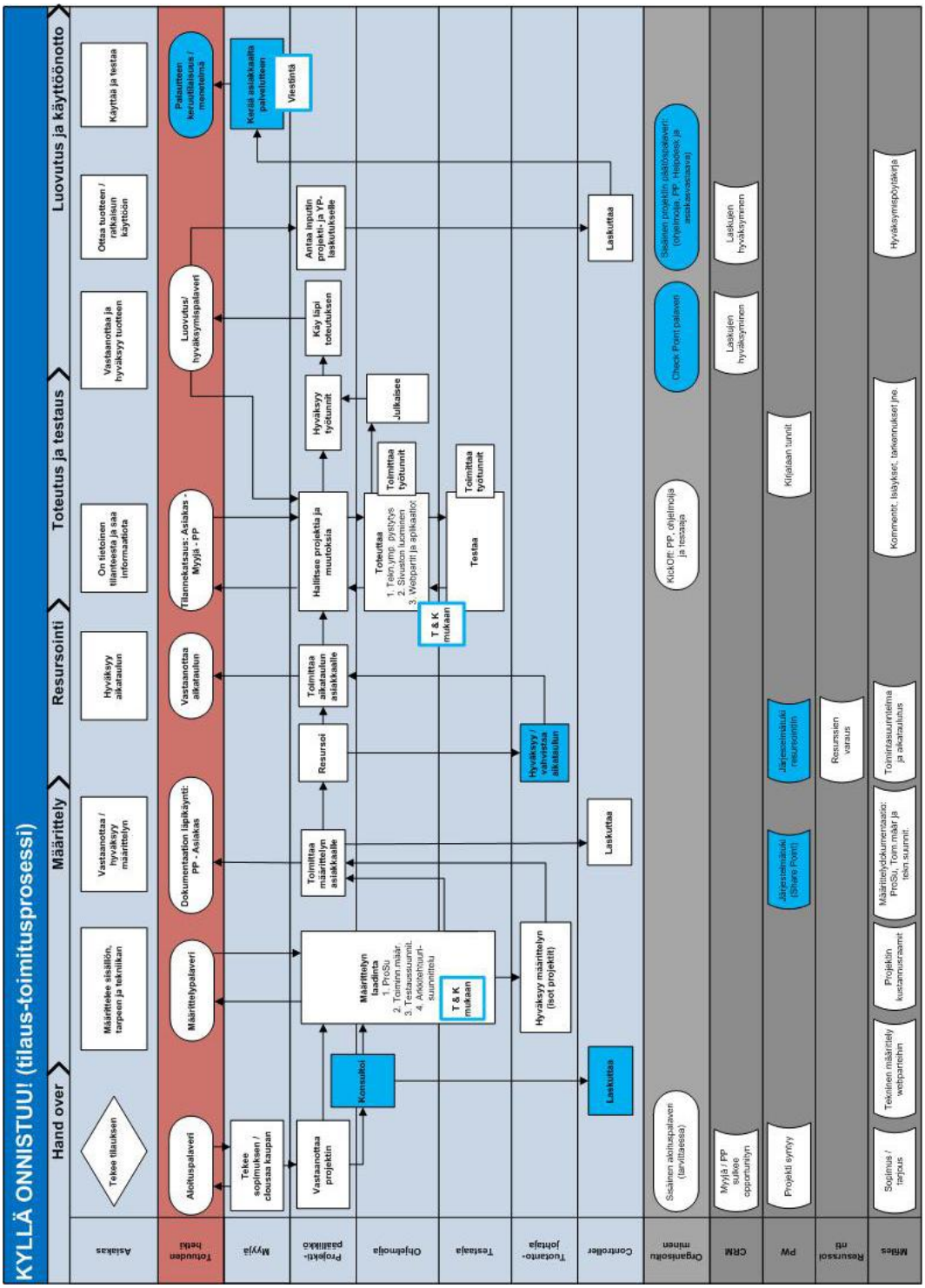
1. Testaus mukana määrittelyvaiheessa tutustumassa projektiin ja toteutukseen. Tässä vaiheessa testaaja voi tehdä testisuunnitelman. Ensimmäisen ja toisen vaiheen välissä testaaja voi tehdä töitä toisen projektin parissa.
2. Testaus mukana toiminnan suunnittelussa ja tarkemmassa suunnittelussa antamassa mielipiteitä. Tässä vaiheessa testaaja tekee tarkempia

testitapauksia testisuunnitelmaan. Testitapauksien pohjalta voidaan luoda testiraporttipohjat, mutta olen havainnut, että raporttien teko on helpompaa testausvaiheessa.

3. Tässä vaiheessa testaaja toteuttaa testiraporttipohjat, suorittaa testauksen ja kirjaa ylös virheet ja tekee niistä virheraportit.
4. Tässä siirrytään hyväksymistestaukseen, jossa testaaja ja asiakas kommunikoivat keskenään ja vertaavat tuloksiaan. Testaaja lähettää asiakkaalle luomansa raporttipohjat, ja asiakas joko testaa niiden pohjalta tai luo omat raportit.

Regressiotestaukseen varataan tarpeeksi aikaa ja projektin aikataulua siirretään tarpeen mukaan. Ohjelmoijat ovat olleet aktiivisempia testauksen suhteen, eikä enää niin monta virhettä pääse läpi testaajalle asti. Hyväksymistestaus suoritetaan, raportteja ja dokumentaatiota laaditaan ja asiakkaiden kanssa kommunikoidaan paljon enemmän kuin ennen.

Visualweb Oy on vuonna 2009 määritellyt prosessejaan tarkemmin, ja ne on saatu kuvattua varsin hyvin. Testaus on otettu huomioon Visualweb Oy :n tilaus-toimitusprosessissa Kyllä onnistuu!. (Kuva 1)



Kuva 1: Tilaus-toimitusprosessi.

9. TESTAUKSEN PROSESSIN TARKENTUMINEN

9.1 Testiympäristön hahmottaminen

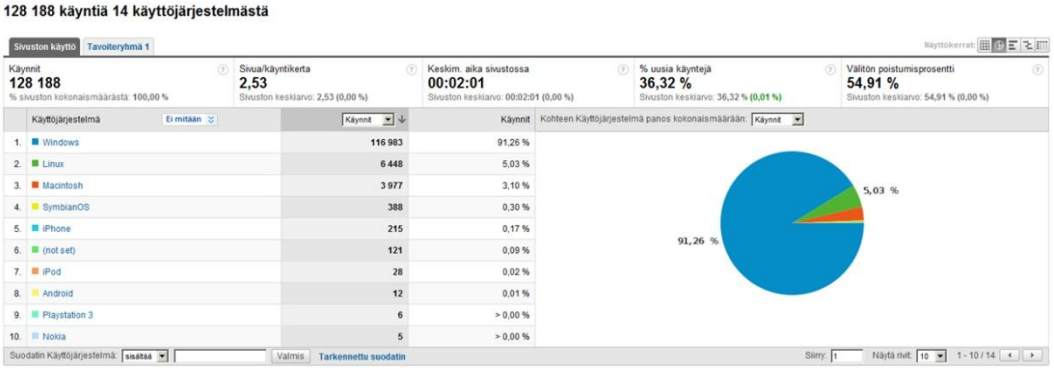
Testiympäristön hahmottamisessa olen käyttänyt apunani eri asiakkaiden statistiikkoja. Hyvänä apuna ovat olleet selain-, käyttöjärjestelmä- ja resoluutiostatistiikat. Esimerkkinä erään keskisuuren kaupungin kuukausistatistiikkoja (Kuva 2 - 4). Itse Service-järjestelmä toimii ainoastaan Internet Explorer 6 tai sitä uudemmilla versioilla.

Kuva 2: Statistiikka selaimista.

Selaimet on esitetty taulukossa (Taulukko 1) näkyvämmin, ja siitä näkee, että käytetyimmät selaimet ovat Internet Explorer, Firefox, Safari, Chrome ja Opera. Suurin osa käyttäjistä käyttää kuitenkin Internet Explorerin ja Firefoxin eri versioita, näiden käyttöprosentti yhteensä on yli 90%.

Taulukko 1. Statistiikka selaimista

	Selain	**Käynnit**	**Käynnit %**
1.	Internet Explorer	80 256	62,61 %
2.	Firefox	39 459	30,78 %
3.	Safari	3 404	2,66 %
4.	Chrome	3 077	2,40 %
5.	Opera	1 616	1,26 %

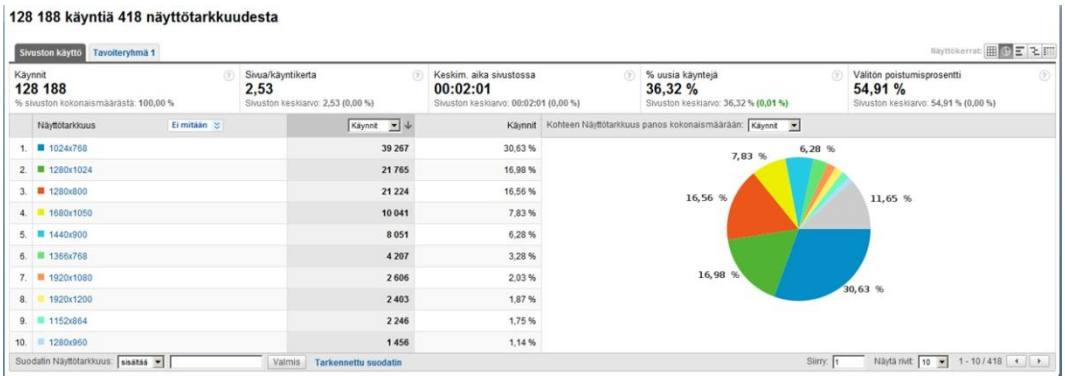


Kuva 3: Statistiikka käyttöjärjestelmistä.

Käyttöjärjestelmät on esitetty taulukossa (Taulukko 2) näkyvämmin, ja siitä näkee, että käytetyimmät käyttöjärjestelmät ovat Windows, Linux, Macintosh, Symbian OS ja iPhone. Suurin osa käyttäjistä käyttää kuitenkin Windows- ja Linux-järjestelmiä, näiden käyttöprosentti yhteensä on yli 95%.

Taulukko 2. Statistiikka selaimista

	Käyttöjärjestelmä	**Käynnit**	**Käynnit %**
1.	Windows	116 983	91,26
2.	Linux	6 448	5,03
3.	Macintosh	3 977	3,10
4.	Symbian OS	388	0,30
5.	iPhone	215	0,17



Kuva 4: Statistiikka näytön resoluutioista.

Käyttöjärjestelmät on esitetty taulukossa (Taulukko 3) näkyvämmin, ja siitä näkee, että käytetyimmät näyttöresoluutiot ovat 1024 x 768, 1280 x 1024, 1280 x

800, 1680 x 1050 ja 1440 x 900. Suurin osa käyttäjistä käyttää kuitenkin 1024 x 768, 1280 x 1024 ja 1280 x 800 resoluutiota, näiden käyttöprosentti yhteensä on yli 60%.

Taulukko 3. Statistiikka näytön resoluutioista

	Näyttöresoluutio	**Käynnit**	**Käynnit %**
1.	1024 x 768	39 267	30,63
2.	1280 x 1024	21 765	16,98
3.	1280 x 800	21 224	16,56
4.	1680 x 1050	10 041	7,83
5.	1440 x 900	8 051	6,28

Näitä tietoja apuna käyttäen olen määritellyt Visualweb Oy :lle seuraavat testiympäristöt:

- Käyttöjärjestelmät ja selaimet:
    - Windows XP, Vista ja Win7
        - Internet Explorer 7 ja 8.X versiot
        - Firefox Mozilla 1.5, 2 ja 3.X versiot
        - Opera 9 ja 10.X versiot
    - Macintosh-järjestelmä
        - Safari
        - Firefox Mozilla 2, 3.X versiot
        - Opera 10.X versio
- Resoluutiot:
    - 1024 x 768
    - 1280 x 1024
    - 1280 x 800
    - 1600 x 1024
    - 1920 x 1080

Internet Explorer 6 -selainta käytetään kuitenkin vielä paljon. Kunnissa, kaupungeissa sekä isoissa organisaatioissa saattaa olla muita ohjelmistoja, jotka jarruttavat uusimpien selainversioiden käyttöönottoa. Netscape-selain ei ole testilistalla, sillä sen ylläpito ja kehitys ovat päättyneet. Chrome-selaimen

mukaanotosta testilistalle olen laittanut ehdotuksen tuotantojohtajalle, ja se odottaa hyväksyntää. Asiakkaan tarpeet otetaan aina huomioon, ja selainlistaa muutetaan asiakkaan toiveiden mukaan. Yleisesti toimintatapana on, että jos selaimia lisätään listalle, siitä laskutetaan asiakasta erikseen, sillä se lisää työmäärää ja –aikaa. Jos selaimia vaihdetaan, lisälaskutusta ei synny. Jos selaimia poistetaan listalta, testauksen työmäärä pienenee huomattavasti, ja se otetaan huomioon projektin kokonaisajassa. Kaikki toiminnot eivät toimi kaikilla selaimilla samankaltaisesti tai ollenkaan, tämä tulee tehdä asiakkaalle selväksi, mikäli tällainen toiminto tulee määrittelyssä esille.

Testiympäristön määrittelyssä otin myös huomioon ohjelmoijien mielipiteet ja käytin hyödykseni asiakkaiden viimeisimpiä toiveita. Määrittelyn hyväksytin tuotantojohtajalla ja hallituksella. Tällä hetkellä testiselainlista liitetään jokaisen projektin sopimukseen. Kun uusia selaimia tulee markkinoille, ja niillä ilmenee virheitä vanhalla sivustolla, asiakasta laskutetaan näiden virheiden korjauksista. Jos projekti on saatettu loppuun, ja siihen tehdään myöhemmässä vaiheessa laajennusta tai muutosta, sopimukseen liitetään aina uusin selainlista, eikä vanha enää päde muutoskohteisiin.

Osa testiympäristöistä on asennettu VmWarelle, yksi on kannettavalla tietokoneella ja yksi ympäristö on Macintosh-pohjainen. VmWare Workstation on ohjelmisto virtuaalitietokoneiden luontiin ja ajamiseen. Sillä voidaan ajaa yhden fyysisen tietokoneen päällä yhtä tai useampaa virtuaalista x86-tietokonetta. (Wikipedia) Mikään käytettävistä selaimista ei ole Stand Alone -versio, sillä ne käyttäytyvät eri tavalla kuin normaalit versiot.

9.2 Web-perusprojektin perusosat

Pienempi, perus web-projekti Visualweb Oy :ssä sisältää vain www-sivuston. Sivusto rakennetaan Visualweb Oy :n alustalle ja sivupohjat ohjelmoidaan Service-järjestelmään. Service on Visualweb Oy :n kehittämä sisällönhallintatyökalu, jolla asiakkaat voivat itse helposti ja ilman ohjelmointiosaamista ylläpitää sivustoaan. Sivusto siis rakennetaan Service-järjestelmään, johon tehdään ulkoasukuvien mukaiset sivupohjat.

Toiminnallisuudet ohjelmoidaan määrittelyn mukaisesti ja liitetään sivupohjiin. Yleensä sivupohja sisältää yhden tai useamman sisältöalueen, menu-ominaisuuden, bannerialueen ja kovakoodattuja linkkejä. Sivupohjiin voidaan liittää erillisiä ominaisuuksia, kuten tietokannasta tulevia uutisia tai lomakkeita. Sivupohjien määrä vaihtelee asiakkaan tarpeiden mukaan, yleensä sivupohjia on pienemmissä projekteissa kaksi tai kolme; yksi sivupohja on etusivun sivupohja ja muut alasivujen sivupohjia. Sivustolla on yleensä vaakatasoinen menu ja alasivuilla myös pystysuuntainen menu, joka näyttää alasivut. Menutasot määritellään asiakkaan tarpeiden mukaan. Sivustolla on usein uutisia etusivulla, uutiset asiakas lisää helposti Servicessä tietokannan hallintaan, ja uutiset näkyvät niille määrätyillä alueilla sivupohjissa. Sivustolla on yleensä palautelomake sekä sivukartta.

Vaativammat projektit sisältävät usein monta lomaketta, monta tietokantaa, käyttäjien rekisteröitymistä ja kirjautumista. Esimerkkejä vaativimmista projekteista, joita olen testannut, ovat Martela, Urheiluhallit ja Finfood. Martelalla on monta kieliversiota tuotesivustosta, kieliversiot poikkeavat jollakin tapaa toisistaan . Urheiluhalleilla on käyttäjien rekisteröitymistä sekä liikuntakurssien varaamista ja myymistä. Finfood-projektissa luotiin monta pienempää alasivustoa sekä pääsivustot, joiden toiminnallisuudet olivat yhteyksissä toisiinsa. Pääsivuston menun kautta pääsi alasivustoille, ja alasivustoilla oli erilaisia uutis- ja tiedoteominaisuuksia sekä reseptihakuja. Projektin toteutus kesti noin puolitoista vuotta, ja testaus oli erittäin hankalaa, koska sivuston määrittely muuttui monta kertaa.

9.3 Web-projektin testauksen eri osa-alueet

Visualweb Oy :ssä tehdään monta erityyppistä testausta ja malleja on useita, sillä yrityksessä tapahtuu järjestelmäkehitystä sekä web-sovellusten ohjelmointia. Web-projektien testaus koostuu pääasiassa moduulitestauksesta, integraatiotestauksesta, järjestelmätestauksesta sekä hyväksymistestauksesta.

Ohjelmoijat suorittavat pääasiassa moduulitestausta ja integraatiotestausta. Testaaja puolestaan tulee apuun integraatiotestauksessa ja suorittaa

järjestelmätestauksen, käyttöliittymätestauksen, selaintestauksen ja eri toiminnallisuuksien testauksen erilaisilla arvoilla. Ei-toiminnalliseen testaukseen kuuluu lähinnä turvallisuus- ja käyttöoikeustestaus. Suorituskyvyn testaus suoritetaan asiakkaan pyynnöstä.

Moduulitestauksessa katsotaan sovelluksen osan toimivuutta ennen sen yhdistämistä muihin sovelluksen osiin ja integraatiotestauksessa yhdistetään moduuleita ja selvitetään niiden toimivuus yhdessä. Uusia moduuleja testataan esimerkiksi antamalla URL-parametreinä oikeita/virheellisiä syötteitä. Jos kyseessä on muokattu moduuli, testauksen tarve riippuu muutosten määrästä. Muokatuissa moduuleissa testataan myös muutoksista riippumattomia moduulien osioita eli luokkia, ominaisuuksia, funktioita, metodeita yms.. Rajapinnat ovat tärkeitä testauksen kohteita. Integraatiotestauksessa käytetään yleensä Bottom up -menetelmää, jossa testataan alimmat moduulit ensin, ja sitten pikkuhiljaa lisätään moduuleja, joskus käytetään myös Bottom up- ja Top down- menetelmien sekoitusta.

Koska liikutaan web-maailmassa, on käytössä erilaista koodia, kuten esimerkiksi HTML, CSS, JavaScript sekä ASPX. Testauksen apuna käytetään erilaisia koodi- ja CSS-validaattoreita, Mozilla-selaimen työkaluja, lokiin kirjoittamista, Tracing-ominaisuutta, Visual Studio Debuggeria, SQL Server Profileria ja Query Analyzeria. Kaikki selaimet eivät kuitenkaan tue niin sanottua standardoitua koodia, vaan vaativat oman standardinsa mukaisen koodauksen. Eräällä asiakkaallamme oli tilanne, että w3c-validaattori antoi virheitä, jotka asiakas vaati korjaamaan, ja korjauksen jälkeen kaikki toiminnallisuudet eivät toimineet enää Firefox- ja/tai Internet Explorer -selaimilla.

Perusprojektien testauksessa testataan ihan ensimmäiseksi sivupohjat, eli testataan toimivatko ne oikein Servicessä. Sivupohjista testataan sisältöalueet ja tietokannan kautta tulevien tietojen näkyminen niille tarkoitetuilla alueilla. Tarkistetaan että tekstityypit ovat oikeita, määrittelyn mukaisia, sisältöalueita on määrittelyn mukainen määrä ja että eri toiminnallisuudet toimivat oikein. Sivustoon tehdään testirakenne ja syötetään testidataa selaintestausta varten.

Vaativimmissa projekteissa tehdään myöskin niin sanottu Service-testaus, ja sen jälkeen, riippuen projektin määrittelystä, tehdään testaussuunnitelma (liite 4). Määrittelyn ja toteutuksen pohjalta tehdään testitapaukset, ja testiraportti (liite 5) luodaan joko ennen testausta tai testauksen aikana. Kun sivusto on valmis testaukseen, se julkaistaan Visualwebin verkkoon, sivustoon pääsee käsiksi lisäämällä palvelimen IP-osoitteen oman tietokoneensa HOST-tiedostoon.

Testaaja aloittaa selaintestauksen yleensä vertaamalla sivustoa piirrettyihin ulkoasukuviin, eli niin sanottuihin leiskoihin. Tulostuksen ulkoasu on myöskin määritetty CSS-tiedostoissa, ja tulostus tulee myöskin testata kaikilla selaimilla. Useimmin kaikki sivuston ominaisuudet eivät ole samanaikaisesti valmiit, joten testaus suoritetaan vaiheissa. Sivuston ulkoasu ja määritellyt ominaisuudet selaintestataan erilaisissa testiympäristöissä. Testauksessa kiinnitetään huomiota paitsi ulkoasuun, myös toiminnallisuuksiin. Sivuston kaikki lomakkeet testataan siten, että kenttiin syötetään oikeita/virheellisiä arvoja; pakolliset kentät jätetään esimerkiksi täyttämättä ja katsotaan millaisia virheilmoituksia järjestelmä palauttaa vai aiheutuuko jostakin virhe. Jos lomakkeen tiedot lähetetään säköpostiin tai talletetaan kantaan, katsotaan että tiedot välittyvät oikein lähetyksen jälkeen. Virheitä yritetään aiheuttaa esimerkiksi erikoismerkeillä kentissä tai liian pitkillä syötteillä. Jos kyseessä on web-kauppa, jossa on osto-ominaisuus, web-kauppa käydään erittäin tarkasti läpi, ja tehdään valmiiksi testitapaukset. Virheitä voi ilmetä jo kaupan puolella, siirryttäessä omaan verkkopankkiin tai maksutilanteessa. Kaikki tilanteet käydään läpi ja katsotaan miten järjestelmä käyttäytyy. Intraneteissä tärkeintä on tietosuoja, eli tarkistetaan, että ulkopuoliset eivät pääse intranettiin ja intranetin tiedostoihin mitenkään käsiksi. Vaativimmissa projekteissa tarvitaan joskus ohjelmoijan apua testien tekemiseen, varsinkin kun jotkut tiedot haetaan kolmannen osapuolen järjestelmästä, kuten Urheiluhalleilla tehdään.

Testauksen päätyttyä ja virheraporttien (liite 6) valmistuttua sivusto siirtyy korjaukseen, mikäli virheitä on löytynyt. Jos testaus tehdään vaiheissa, testataan ensin yksi järjestelmän osa, ja se lähetetään korjattavaksi. Samalla kun ensimmäistä osaa korjataan, testataan toinen järjestelmäosa jne.. Korjauksien

jälkeen testaaja suorittaa regressiotestauksen, testauksen kohteina ovat korjatut kohdat, mutta laajempi testaus tulee tehdä, jotta varmistutaan, etteivät muut järjestelmän osat vioittuneet korjauksesta. Regressiotestaus suoritetaan niin monta kertaa kuin tarve vaatii, kaikki virheraportin virheet tulee olla korjattu.

Testausraportti siirtyy asiakkaalle, kun testaaja on todennut kaikkien virheiden olevan korjattuja. Tässä vaiheessa asiakas tekee hyväksymistestauksen. Toiminnallisuuksista riippuen asiakkaalle annetaan 2 päivää – 2 viikkoa testaukseen, jonka jälkeen asiakas palauttaa täytetyn raportin takaisin testaajalle. Asiakkaalla on myös mahdollisuus testata sivustoa yhtä aikaa testaajan kanssa ja tehdä oma testiraportti. Testaajan saatua testiraportti, käy hän sen läpi ja yrittää toistaa asiakkaan löytämiä virheitä. Testaaja ja asiakas kommunikoivat tässä vaiheessa usein. Testaaja siirtää raportin ohjelmoijalle, joka korjaa löydetyt virheet. Regressiotestaus suoritetaan tässä vaiheessa uudestaan, ja kun virheet on korjattu, siirtyy raportti takaisin asiakkaalle. Asiakas joko hyväksyy raportin, ilmoittaa uusista virheistä tai havaitsee että vanhoja ei ole korjattu. Jos asiakkaan testiraportissa on määrittelystä poikkeavia toivomuksia tai muutospyyntöjä, siirtyvät ne muutoksenhallintaprosessiin. Testiraportti saa sisältää vain sivuston virheitä, eli määrittelystä poikkeavia asioita. Kun asiakas on hyväksynyt projektin, voidaan se julkaista. Projekti siirtyy julkaisun jälkeen ylläpitoprosessiin.

9.4 Testauksen tuomat hyödyt yritykselle

Testausprosessin selkeytyminen ja ylipäätään testaus yrityksessä on tuonut paljon hyötyjä. Kun projekti testataan, ja asiakas sen hyväksyy, voidaan se laskuttaa ja myöhemmät muutokset laskutetaan erikseen. Yritys säästää rahaa, sillä virheet korjataan jo ennen sivuston julkaisua, ja myöhempien virheiden löytymisen riski minimoidaan. Koska virheitä on sivustoissa paljon vähemmän kuin ennen, on HelpDesk-puheluiden määrä vähentynyt huomattavasti, ja asiakkaat ovat tyytyväisempiä yrityksen palveluihin.

10. YHTEENVETO

Testausprosessi on pieni, mutta erittäin tärkeä osa projektin prosessia. Testausta ei tulisi vähätellä, ja siihen tulisi panostaa kunnolla. Testauksessa on aina parantamisen varaa, ja sitä tulisikin kehittää jatkuvasti.

Visualweb Oy :ssä testausta sovelletaan uusiin järjestelmiin, uusia testiympäristöjä rakennetaan ja prosessia kehitetään koko ajan. Testauksen tapa toimia ei aina mene niin kuin pitäisi; välillä jää asioita tekemättä jossakin vaiheessa projektin prosessia, mikä vaikeuttaa kaikkia myöhempiä projektin vaiheita. Kaikki eivät ole vielä yrityksessä omaksuneet tapaa toimia testauksen suhteen, joskus aikaa ei ole tarpeeksi, regressiotestausta ei suoriteta tai kiireellisen aikataulun johdosta testaus ei ole tarpeeksi kattava.

Testaus ja sen hyvä suorittaminen on kuitenkin avannut yrityksen silmät, ja testausta arvostetaan ja siihen panostetaan nyt enemmän kuin ennen.

Itse opinnäytetyön tekeminen on ollut vaativa ja pitkä prosessi. Olin uusi testauksen alalla, eikä kaikkia työkaluja ollut valmiina. Testauksen suunnittelu, testaussuunnitelmien sekä raporttien tekeminen ja hiominen vaati aikaa ja kärsivällisyyttä. Olen oppinut paljon testauksesta, web-maailmasta sekä projektien johtamisesta. Testaustyöni ei rajoitu vain asiakassivustojen testaamiseen, vaan osallistun myös tuotekehitystestaukseen, mikä on vaativampi osa-alue. Työ sujuu nykyään onneksi paljon nopeammin, ja pystyn välttämään ihan perusvirheitä, joita alussa tuli tehtyä useammin.

LÄHTEET

Haikala I. ja J. Märijärvi (2001), Ohjelmistotuotanto. 7. uudistettu painos, Pieksämäki 2001.

Ketterät käytännöt [online] - http://www.ketteratkaytannot.fi/fi-FI/Ketteryys/Periaatteet/

Ron Patton (2002), Software Testing. Second edition.

Tieturi Oy – Testauksen valmennusohjelman materiaali (2007).

Tieturi Oy – Web-sovellusten testauskurssin materiaali (2007).

Wikipedia [online]- http://fi.wikipedia.org/wiki/VMware#VMware_Workstation

LIITELUETTELO

9.2.2010 22:01:04

Test Report

Webserver Load Performance Stress Test

Test Type: RAMP (run test for 5 minutes)

User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks

Summary Log

```
** Test Logfile by Webserver Stress Tool 7.2.2.258 Enterprise
Edition (1 User License) **
© 1998-2008 Paessler AG, http://www.paessler.com

Test run on 9.2.2010 22:01:04

** Project and Scenario Comments, Operator **


Results of period #1 (from 2 sec to 7 sec ):
************************************************************
Completed Clicks: 40 with 0 Errors (=0,00%)
Average Click Time for 44 Users: 636 ms
Successful clicks per Second: 7,96 (equals 28 644,98 Clicks
per Hour)
Results of period #2 (from 7 sec to 12 sec ):
************************************************************
Completed Clicks: 40 with 0 Errors (=0,00%)
Average Click Time for 86 Users: 352 ms
Successful clicks per Second: 7,91 (equals 28 471,76 Clicks
per Hour)
Results of period #3 (from 12 sec to 17 sec ):
************************************************************
Completed Clicks: 74 with 0 Errors (=0,00%)
Average Click Time for 128 Users: 472 ms
Successful clicks per Second: 14,65 (equals 52 754,24 Clicks
per Hour)
Results of period #4 (from 17 sec to 22 sec ):
************************************************************
Completed Clicks: 82 with 0 Errors (=0,00%)
Average Click Time for 171 Users: 441 ms
Successful clicks per Second: 16,05 (equals 57 763,49 Clicks
per Hour)
Results of period #5 (from 22 sec to 27 sec ):
************************************************************
Completed Clicks: 117 with 0 Errors (=0,00%)
Average Click Time for 214 Users: 579 ms
Successful clicks per Second: 22,59 (equals 81 322,05 Clicks
per Hour)
Results of period #6 (from 27 sec to 32 sec ):
************************************************************
Completed Clicks: 123 with 0 Errors (=0,00%)
Average Click Time for 257 Users: 618 ms
Successful clicks per Second: 23,75 (equals 85 482,23 Clicks
per Hour)
Results of period #7 (from 32 sec to 38 sec ):
```

.
.
.

```
Results of complete test
************************************************************

** Results per URL for complete test **

URL#1 (): Average Click Time 2 182 ms, 20 794 Clicks, 644
Errors

Total Number of Clicks: 20 794 (644 Errors)
Average Click Time of all URLs: 2 114 ms


!! Glossary:
!! Click: A simulated mouse click of a user sending a request
(one of the
URLs from the URL list) to the server and immediately
requesting any
necessary redirects, frames and images (if enabled).
!! Request: A HTTP request sent to the server regardless of an
answer.
!! Hit: A completed HTTP request (i.e. sent to the server and
answered
completely). Hits can be the PAGE request of a "click" or its
frames, images
etc.
!! Time for DNS: Time to resolve a URL's domain name using the
client
system's current DNS server.
!! Time to connect: Time to set up a connection to the server.
!! Time to first byte (TFB): Time between initiating a request
and receiving
the first byte of data from the server.
!! Click Time: The time a user had to wait until his "click"
was finished
(including redirections/frames/images etc.).
!! User Bandwidth: The bandwidth a user was able to achieve.
!! Sent Requests: Number of requests sent to the server during
a period.
!! Received Requests: Number of answers received from the
server during a
period.
```

Results per User

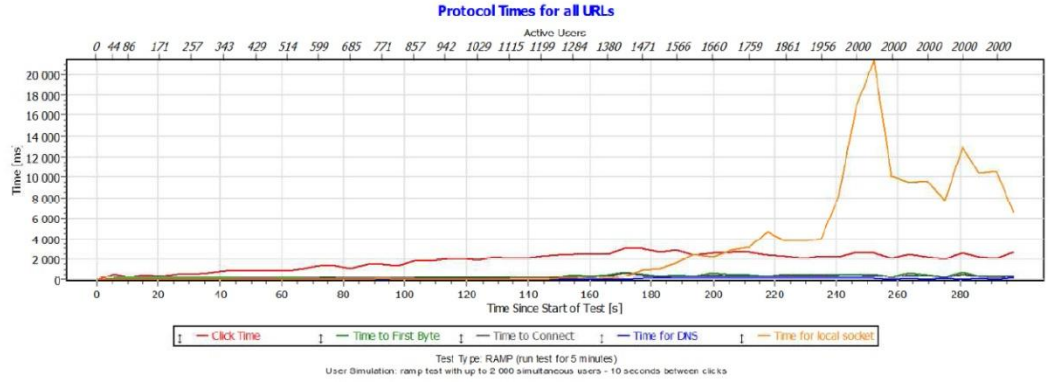
User No.	Clicks	Hits	Errors	Avg. Click Time [ms]	Bytes	kbit/s	Cookies
1	22	22	1	2 455	1 291 985	191,34	
2	22	22	1	2 637	1 291 647	178,13	
3	23	22	0	1 831	1 353 154	268,68	
4	21	21	1	2 172	1 209 494	212,10	
5	22	22	1	1 952	1 291 647	240,59	
6	24	24	1	1 768	1 414 661	266,64	
7	22	22	1	2 423	1 265 054	189,88	
8	20	20	1	2 597	1 134 782	174,77	
9	22	22	1	2 718	1 288 244	172,35	
10	23	22	0	1 865	1 333 143	259,97	
11	21	21	0	2 573	1 281 503	189,72	
12	21	20	0	2 083	1 208 745	232,15	
13	21	20	0	2 221	1 230 140	221,50	

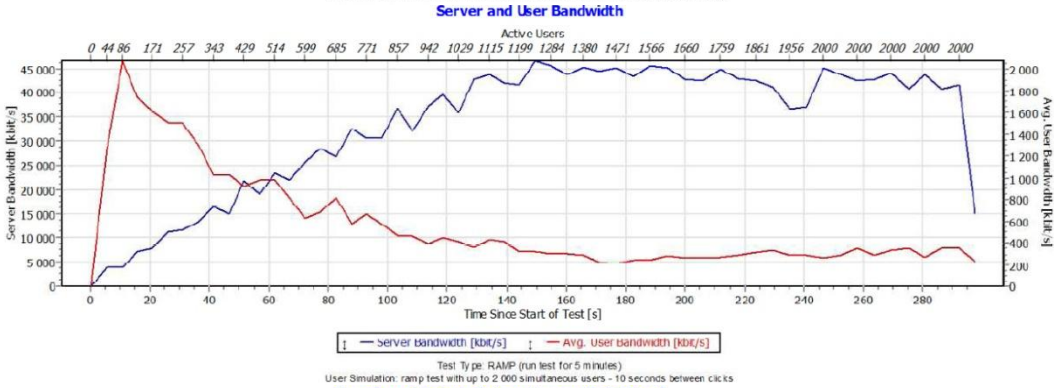
.
.
.

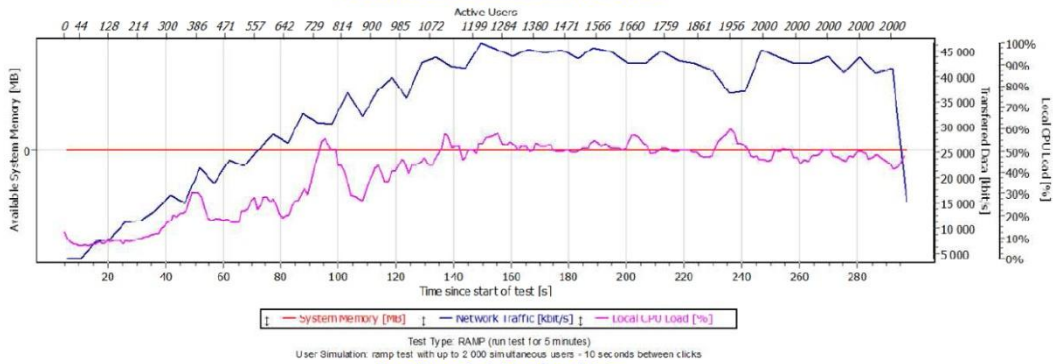
1989	3	3	1	4 794	123 352	68,62	
1990	3	3	1	6 534	123 014	50,21	
1991	3	2	0	5 677	123 014	86,67	
1992	3	3	0	4 729	180 548	101,82	
1993	3	3	1	4 954	123 014	66,22	
1994	3	2	0	5 926	108 539	73,27	
1995	3	3	1	4 354	123 763	75,80	
1996	3	3	1	4 418	123 763	74,70	
1997	3	3	1	6 389	123 352	51,49	
1998	3	3	1	4 530	123 763	72,86	
1999	3	3	1	6 462	123 014	50,76	
2000	3	3	1	5 877	123 352	55,97	

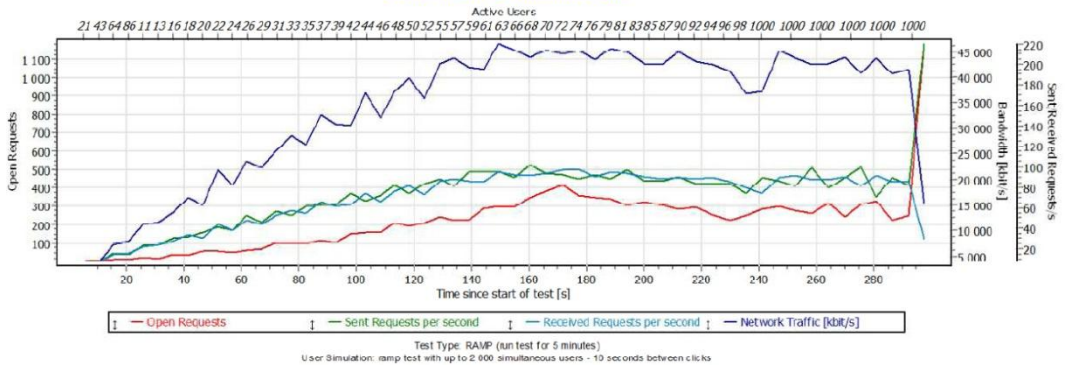
Results per URL

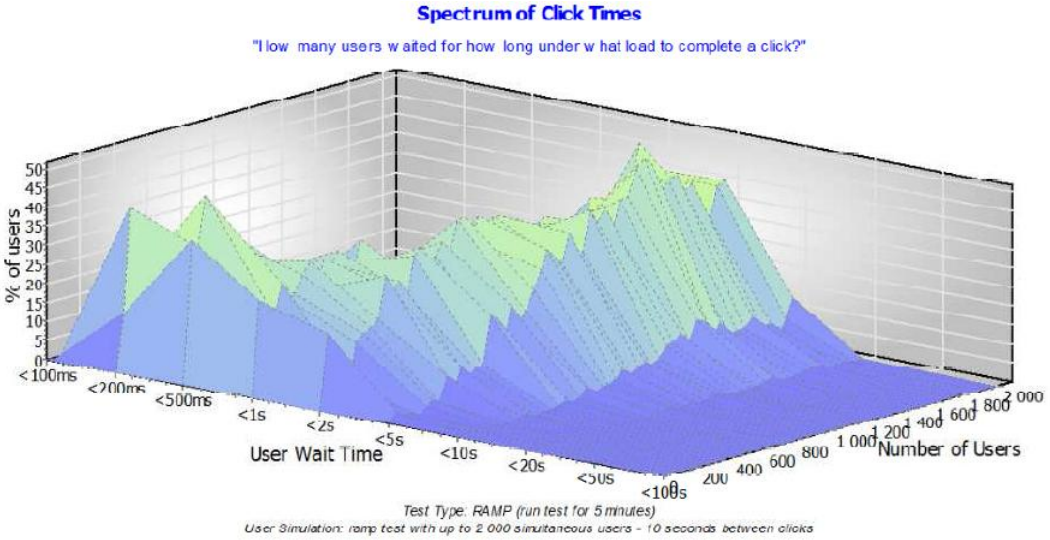
URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		20 794	644	3,10	43 957 348	2 182

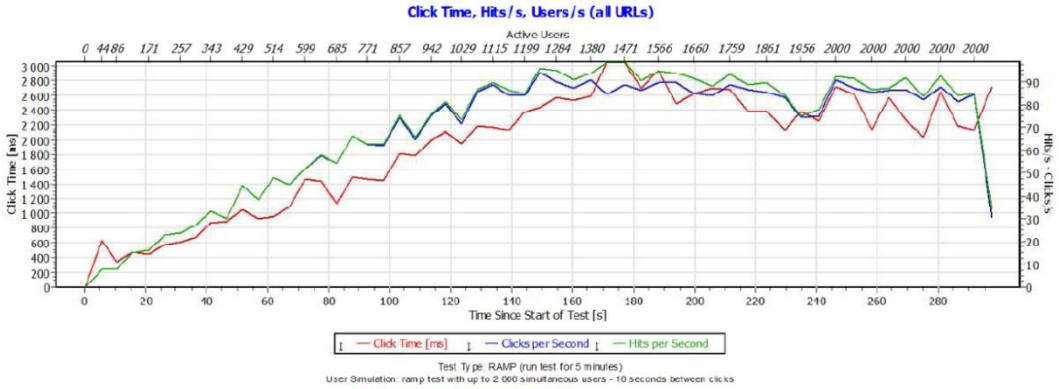

Protocol Times for all URLs
Active Users
Time [ms]
Time Since Start of Test [s]
Click Time
Time to First Byte
Time to Connect
Time for DNS
Time for local socket
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks

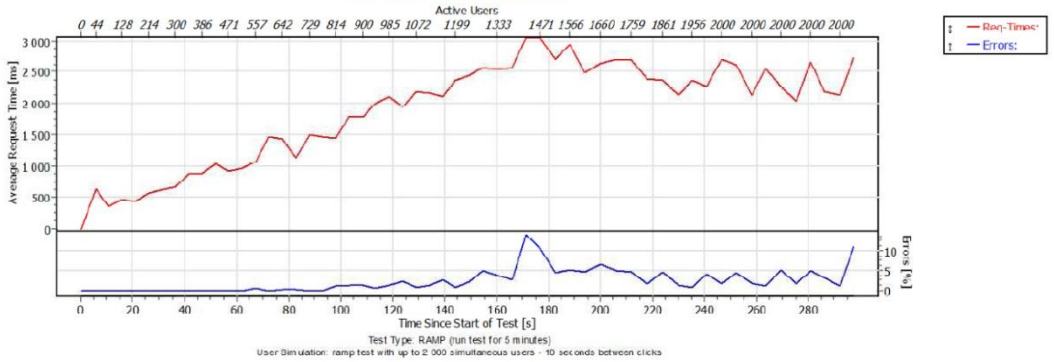

Server and User Bandwidth
Active Users
Server Bandwidth [kbit/s]
Avg. User Bandwidth [kbit/s]
Time Since Start of Test [s]
Server Bandwidth [kbit/s]
Avg. User Bandwidth [kbit/s]
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks


Transferred Data & System Memory & CPU Load
Active Users
Available System Memory [MB]
Transferred Data [kbit/s]
Local CPU Load [%]
Time since start of test [s]
System Memory [MB]
Network Traffic [kbit/s]
Local CPU Load [%]
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks


Open Requests & Transferred Data
Active Users
Open Requests
Bandwidth [kbit/s]
Sent/Received Requests/s
Time since start of test [s]
Open Requests
Sent Requests per second
Received Requests per second
Network Traffic [kbit/s]
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks


Spectrum of Click Times
"How many users waited for how long under what load to complete a click?"
% of users
User Wait Time
Number of Users
<100ms
<200ms
<500ms
<1s
<2s
<5s
<10s
<20s
<50s
<100s
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks


Click Time, Hits/s, Users/s (all URLs)
Active Users
Click Time [ms]
Hits/s - Clicks/s
Time Since Start of Test [s]
Click Time [ms]
Clicks per Second
Hits per Second
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks


Click Times and Errors (per URL)
Active Users
Average Request Time [ms]
Errors [%]
Time Since Start of Test [s]
Req-Times
Errors
Test Type: RAMP (run test for 5 minutes)
User Simulation: ramp test with up to 2 000 simultaneous users - 10 seconds between clicks

Pieturi

Todistus

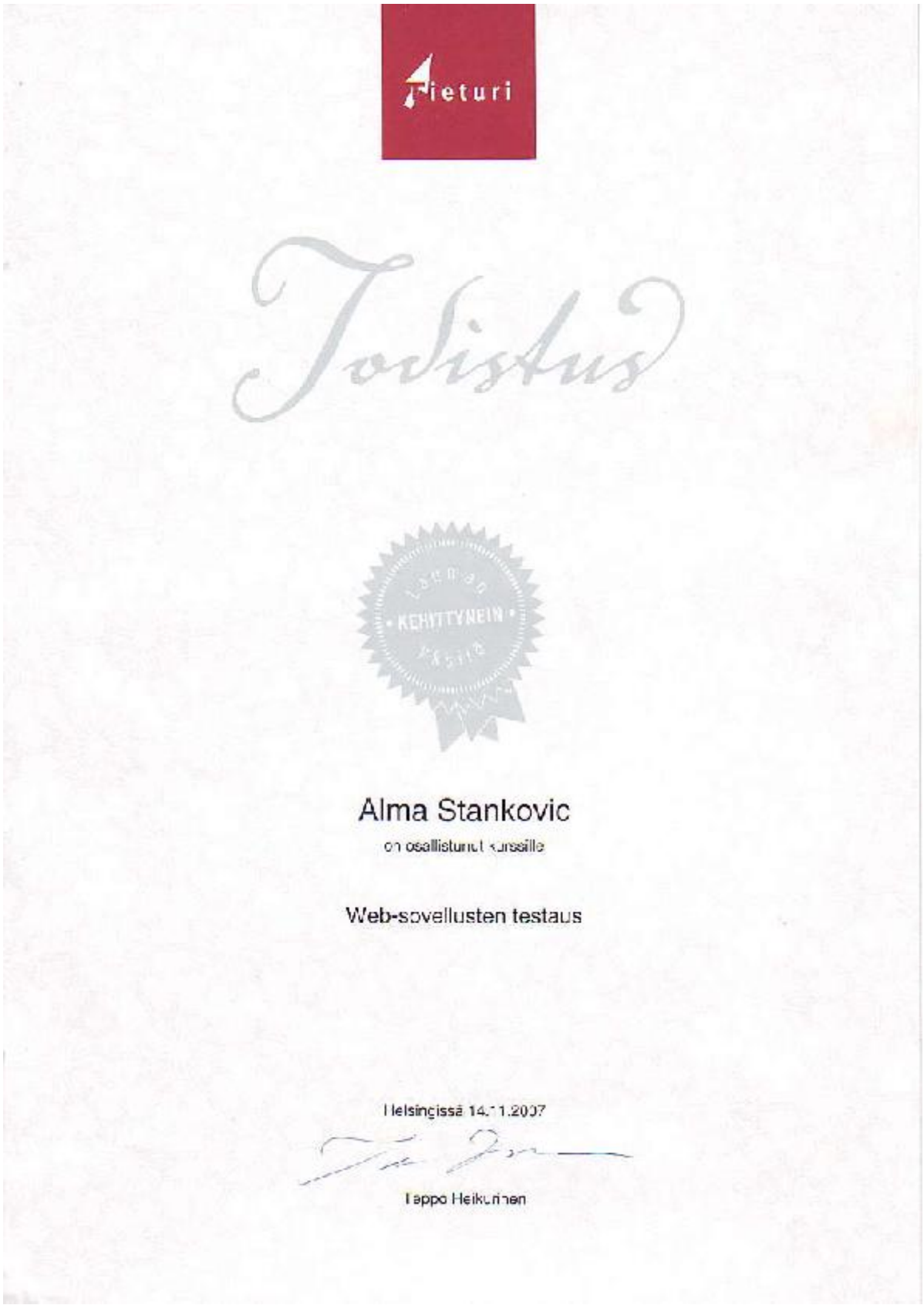
Alma Stankovic

on onnistuneesti suorittanut

Testauksen valmennusohjelman lopputentin 21.3.2007
Lopputentin arvosana: Kiitettävä

Helsingissä **4.4.2007**

Teppo Heikurinen


Tieturi
Todistus
KEHITTYNEIN
Alma Stankovic
on osallistunut kurssille
Web-sovellusten testaus
Helsingissä 14.11.2007
Teppo Heikkinen

Alma Stankovic

26.5.2010

Yritys-hanke

Objektienhallinta - Testaussuunnitelma

Versio 0.3

Alma Stankovic

26.5.2010

Sisältö

Alma Stankovic 26.5.2010

1. Yleistä

1.1. Dokumentin tarkoitus

Tämä dokumentti kuvaa Yritys-hankkeen testauksen suunnittelua teknisellä tasolla.

1.2. Tavoitteet

Testauksen tavoitteena on pyrkiä mittaamaan ja parantamaan toteuttavan sovelluksen laatua. Testaus tapahtuu yleensä monella tasolla ns. V-mallin mukaisesti. V-mallissa testaus jaetaan moduulitestaukseen (yksikkötestaus), järjestelmätestaukseen ja hyväksymisteastaukseen. Kaikki testauksessa löytyneet virheet pyritään korjaamaan ja ohjelmisto testaamaan uudelleen.

1.3. Kohdeyleisö

Tämän dokumentin kohdeyleisö on Yritys-hankkeen ohjausryhmä, määrittelijät, projektipäälliköt, toteuttajat ja testaajat.

1.4. Termit

Termi	Selitys
Sharepoint	Microsoftin selainpohjainen tuote, joka tarjoaa välineet asiakirjojen hallintaan ja selainpohjaisen tiedonhallintasivuston rakentamiseen.
MS Exchange	Microsoftin sähköpostipalvelin (http://en.wikipedia.org/wiki/Microsoft_Exchange_Server)
WSS	Microsoft Windows Server 2003 ja 2008-versioissa oleva ilmainen lisäosa joka tarjoaa verkkoportaalin ominaisuuksia. (http://en.wikipedia.org/wiki/Windows_SharePoint_Services)
MOSS	Microsoft Office Sharepoint Server on WSS-toiminnallisuuksia laajentava kaupallinen tuote. (http://en.wikipedia.org/wiki/Microsoft_Office_SharePoint_Server)
WebPart	Toiminteen tarjoava yksittäinen verkkosivun osa. (http://en.wikipedia.org/wiki/Web_part)

1.5. Versionhallinta

Kirjoittaja	Versio	Muutokset	Tarkastaja
Sna	0.1	Pohja	

Ast	0.2	Testaussuunnitelma	
Ast	0.3	Kappaleen 2.2 päivittäminen vastaamaan projektisuunnitelman kappaletta 5.2	

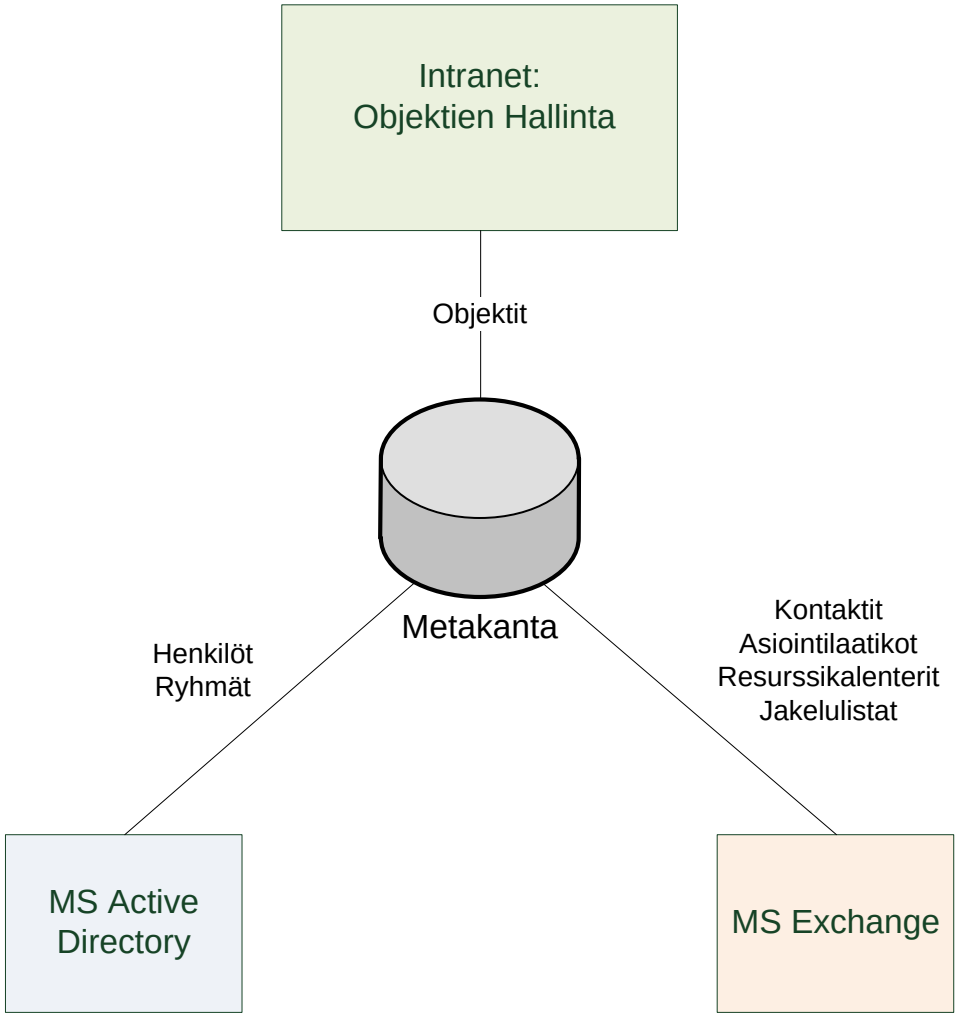
2. Testaussuunnitelma

Tässä luvussa kuvataan Yritys-hankkeen objektienhallintatestaussuunnitelma yleisellä tasolla. Testejä pidetään mahdollisimman kattavana ja toistettavana. Testaussuunnitelma pohjautuu Yritys – Objektien hallinta projektin määrittelyyn ja ulkoasuohjeistuksiin.

2.1. Testattava ohjelmisto

Objektien hallinta on MOSS:n päälle rakennettu palvelu, jonka avulla käyttäjät pystyvät hallinnoimaan MS Exchange- ja AD-objekteja.

Objektien hallinta on osa isompaa kokonaisuutta. Toteutus on osa isompaa IAM ja IdM toteutusta ja hanketta, jonka Yritys-hanke käynnisti.



2.2. Testausstrategia

Tämä luku esittää ohjelmiston testausstrategian testausvaiheittain.

2.2.1. Moduulitestaus

Moduulitestauksessa etsitään vikoja yksittäisistä moduuleista. Moduulitestauksen suorittaa moduulin ohjelmoinut projektiryhmän jäsen. Moduulit testataan lasilaatikkotestauksena Sharepoint testiympäristössä ja asiakkaan testiympäristössä. Lasilaatikkotestauksessa testitapausten valinnassa käytetään hyväksi tietoa moduulin toteutuksesta.

Moduulin testausta varten tehdään tarvittaessa testipedejä (test bed), jolla moduulin toimivuutta voidaan kokeilla. Moduulin toteuttaja vastaa testauksesta ja päättää, milloin moduuli on valmis integroitavaksi muihin moduuleihin. Moduulin toteuttaja vastaa myös moduulitestauksessa havaittujen virheiden korjaamisesta.

2.2.2. Järjestelmätestaus

Järjestelmätestauksessa tarkastelun kohteena on koko järjestelmä ja tuloksia verrataan määrittelydokumentaatioon ja asiakasdokumentaatioon. Järjestelmätestauksessa ei testata järjestelmän ei-toiminnallisia ominaisuuksia. Järjestelmätestauksen suorittaa toimittaja.

2.2.3. Hyväksymistestaus

Hyväksymistestauksen suorittaa asiakas omassa ympäristössään ja toimittaa oman virheraportin toimittajalle, jonka pohjalta tarvittavat korjaukset tehdään.

2.3. Testaustyön tulokset

Testaustyön tuloksena valmistuu testausdokumentti, jossa kuvataan suoritetut testitapaukset ja niiden tulokset.

2.4. Testausympäristö

Testausympäristönä toimii Visualwebin Sharepoint alustalle rakennettu testausympäristö. Toisena testausympäristönä toimii asiakkaan testausympäristö.

2.5. Testauksen aikataulu

Ohjelmiston testaaminen aloitetaan 17.3.2010 ja päättyy 28.3.2010. Testaus siirtyy asiakkaalle hyväksymistestaukseen 28.3.2010.

Alma Stankovic 26.5.2010

3. Järjestelmätestauksen testitapaukset

Tässä luvussa on lueteltu ryhmiteltynä kaikki testitapaukset yksilöllisillä tunnisteilla. Testauksen kulusta raportoitavaan testaudokumenttiin merkitään aina tehdyistä testeistä vastaavien testitapauksien tunnukset.

Järjestelmätestauksen testitapauksessa käytetään mustalaatikkomenetelmää. Mustalaatikkotestauksessa testitapaukset valitaan testattava ohjelman määrittelydokumentaation perusteella, josta saatavat tulokset ovat jo etukäteen tiedossa. Tällä tavoin voidaan testata kaikkien webpartien ja algoritmien yhteistoimintaa ja tulosten oikeellisuutta.

Testitapauksista kirjataan syötteiden arvot ja niistä ohjelmassa aiheutuneet tulosteet vastaavalla tarkkuudella testauslomakkeelle, josta valmistetaan testausraportti.

3.1. Ryhmät ja jakelulistat

3.1.1. Ryhmien ja jakelulistojen listaus

Tarkoitus: Tässä listauksessa käyttäjä näkee listattuna kaikki ryhmät sekä jakelulistat yhdessä näkymässä. Näkymässä näytetään vain statuksella ”1” olevat entryt.
Kuvaus: Listauksessa käyttäjä voi suodattaa ryhmiä ja jakelulistoja. Listauksessa käyttäjä voi järjestää näkymää kunkin kolumnin mukaan. Oletusarvoisesti listaus on järjestetty aakkosjärjestyksessä ryhmän/jakelulistan nimen mukaan.
Odotettu tulos: Listaus suodattaa ryhmiä ja jakelulistoja oikein kriteereiden perusteella.

3.1.2. Ryhmien ja jakelulistojen luominen (RJL)

Lomake-RJL
Tarkoitus: Ryhmien ja jakelulistojen luominen kantaan.
Kuvaus: Tämän lomakkeen avulla käyttäjä pystyy luomaan ryhmän tai jakelulistan. Kantaan kirjoitetaan syötetyt arvot sekä lisäksi arvoja kuten päivittäjä.
Odotettu tulos: Ryhmä tai jakelulista luodaan onnistuneesti tietokantaan.

3.1.3. Ryhmien ja jakelulistojen luominen muokkaus (RJL)

Lomake-RJL
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy muokkaamaan ryhmän tai jakelulistan tietoja.
Kuvaus: Lomakkeelle ladataan ryhmän/jakelulistan tiedot, lomakkeella näytetään eri tiedot riippuen siitä onko muokattava objekti ryhmä vai jakelulista. Kun ryhmä tai jakelulista luodaan generoidaan sille käyttäjätunnus (Ryhma.RY_Kayttajatunnus) kenttään. Generoinnissa käytetään erillistä samaa proseduuria kuin uuden luonnissa.
Odotettu tulos: Muutetut tiedot tallettuvat kantaan ja näkyvät ryhmän/jakelulistan tiedoissa.

3.1.4. Ryhmien ja jakelulistojen haku (RJH)

Lomake-RJH
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy hakemaan ryhmiä ja jakelulistoja
Kuvaus: Kaikki haut kohdistuvat Ryhma –tauluun ja tulokset näytetään hakulomakkeen alla.
Odotettu tulos: Palauttaa hakutulokset oikein vapaasanahaun ja rajauksen mukaan.

3.2. Kontaktit

3.2.1. Listaus ja haku (LH)

Lomake - LH
Tarkoitus: Pystytään hakemaan henkilöitä tietokantaa vasten.
Kuvaus: Haku-ruudut sisältävät tekstilaatikon tai –laatikoita joihin haun ehdot kirjoitetaan. Lomake sisältää painikkeen ”Hae”, joka hakee kaikkia niitä objektityyppejä, jotka ovat määritelty haettavaksi. Lomakkeen avulla voi hakea montaa objektia tai vain yhtä.
Odotettu tulos: Kun haku on suoritettu, listautuvat hakutuloket lomakkeen alle. Listalla näkyy eri tietoja riippuen siitä, mitä haettiin. Listauksessa käyttäjä voi järjestää näkymää kunkin kolumnin mukaan. Oletusarvoisesti listaus on järjestetty aakkosjärjestyksessä kontaktin näyttönimen mukaan.

3.2.2. Kontaktin luominen (KL)

Lomake-KL
Tarkoitus: Uuden kontakitn luominen tietokantaan.
Kuvaus: Tämän lomakkeen avulla käyttäjä pystyy luomaan kontaktin täytettyään ainakin pakolliset kentät. Kun kontakti luodaan generoidaan kontaktille käyttäjätunnus (KN_Kayttajatunnus) kenttään.
Odotettu tulos: Luo kontaktin tietokantaan sekä palauttaa käyttäjän kontaktien listausnäkymään.

3.2.3. Kontaktin muokkaaminen (KM)

Lomake-KM
Tarkoitus: Kontaktin tietojen muutos.
Kuvaus: Tämän lomakkeen avulla käyttäjä pystyy muokkaamaan kontaktin tietoja. Lomakkeelle ladataan kontaktin viimeisimmät tiedot. Lomakkeen tulee ilmoittaa, että sähköpostiosoite on jo käytössä.
Odotettu tulos: Muutetut tiedot tallettuvat kantaan ja näkyvät kontaktin tiedoissa.

3.3. Resurssikalenterien hallinta

3.3.1. Resurssikalenterin listaus ja haku (RLH)

Lomake-RLH
Tarkoitus: Haetaan kalenterista resursseja.
Kuvaus: Resurssikalenteri listausnäkymässä on näkyvillä sekä haku-toiminto että listaus resurssikalentereista. Kaikki haut kohdistuvat Resurssikalenteri -tauluun. Listausnäkymästä voidaan hakea monella hakukriteerillä (AND). Listauksessa käyttäjä voi järjestää näkymää kunkin kolumnin mukaan. Oletusarvoisesti listaus on järjestetty aakkosjärjestyksessä resurssikalenterin näyttönimen mukaan.
Odotettu tulos: Näkymässä näytetään hakukriteerien perusteella vain statuksella 1 olevat tulokset. Kun käyttäjä avaa listaussivun näytetään kaikki resurssikalenterit (sivutettuna) hakuehdot ovat tyhjät.

3.3.2. Resurssikalenterin luominen (RL)

Lomake-RL
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy luomaan uuden resurssikalenterin.
Kuvaus: Tämän lomakkeen avulla käyttäjä pystyy luomaan uuden resurssikalenterin. Kun resurssikalenteri luodaan generoidaan sille käyttäjätunnus (Resurssikalenteri .RK_Kayttajatunnus) kenttään ja sähköpostiosoite (Resurssikalenteri .RK_Sahkoposti) kenttään. Sähköposti muodostuu käyttäjätunnus kentästä (RK_Kayttajatunnus) (merkit pienellä) sekä merkkijonosta ”@ahp.root”
Odotettu tulos: Luo resurssikalenterin tietokantaan sekä palauttaa käyttäjän resurssikalenterien listausnäkymään.

3.3.3. Resurssikalenterin muokkaus ja poisto (RMP)

Lomake-RMP
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy muokkaamaan resurssikalenterin tietoja sekä poistamaan resurssikalenterin.
Kuvaus: Resurssikalenterin tietojen muokkaaminen sekä poistamaan resurssikalenterin. Lomakkeelle ladataan resurssikalenterin viimeisimmät tiedot.
Odotettu tulos: Muokatut tiedot tallettuvat oikein kantaan ja palauttaa käyttäjän resurssien näkymään. Poistaa kalenterin, merkitsee kantaan (RK_Status) nollaksi.

3.4. Asiointilaatikoiden hallinta

3.4.1. Asiointilaatikoiden listaus ja haku (ALH)

Lomake-ALH
Tarkoitus: Asiointilaatikoiden listaus ja haku
Kuvaus: Asiointilaatikoiden listausnäkymässä on näkyvillä sekä haku että listaus asiointilaatikoista. Kun käyttäjä avaa listaussivun näytetään kaikki asiointilaatikot (sivutettuna), hakuehdot ovat tyhjä. Kaikki haut kohdistuvat Organisaatiolaatikko -tauluun. Listausnäkymästä voidaan hakea monella hakukriteerillä (AND). Listauksessa käyttäjä voi järjestää näkymää kunkin kolumnin mukaan. Oletusarvoisesti listaus on järjestetty aakkosjärjestyksessä asiointilaatikon näyttönimen mukaan.
Odotettu tulos: Suorittaa haun ja näyttää samalla sivulla hakutuloksen.

3.4.2. Asiointilaatikoiden luominen(AL)

Lomake-AL
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy luomaan uuden asiointilaatikon.
Kuvaus: Kun asiointilaatikko luodaan generoidaan sille käyttäjätunnus (ORL_Kayttajatunnus) kenttään.Generoinnissa käytetään erillistä proseduuria. Käyttäjätunnus muodostuu merkeistä ”RT” sekä juoksevasta numerosta. Alussa numero on viisimerkkinen mutta sadantuhannen ylittyessä tunnuksen numero osa on kuusimerkkinen jne. Numero kasvaa aina kun uusi käyttäjätunnus luodaan.
Odotettu tulos: Luo asiointilaatikon tietokantaan sekä palauttaa käyttäjän asiointilaatikoiden listausnäkymään.

3.4.3. Asiointilaatikoiden muokkaus ja poisto (AMP)

Lomake-AMP
Tarkoitus: Tämän lomakkeen avulla käyttäjä pystyy muokkaamaan asiointilaatikon tietoja sekä poistamaan asiointilaatikon.
Kuvaus: Lomakkeelle ladataan asiointilaatikon viimeisimmät tiedot.
Odotettu tulos: Tallentaa asiointilaatikon tiedot lomakkeelta tietokantaan sekä palauttaa käyttäjän asiointilaatikoiden listausnäkymään. Poistaa asointilaatikon tietokannasta eli merkitsee statuksen (ORL_Status) nollaksi. Varmistaa käyttäjältä poiston ennen poistoa.

Alma Stankovic

26.5.2010

Yritys-hanke: Objektien hallinta

Kontaktit Testausraportti

Versio 1.0

Alma Stankovic

26.5.2010

Sisältö

Alma Stankovic

26.5.2010

1. Yleistä

1.1. Dokumentin tarkoitus

Tämä dokumentti kuvaa Yritys-hankkeen testauksen teknisellä tasolla. Dokumentti kuvaa objektien hallinnan toiminnallisuuksien testauksen loogisina kokonaisuuksina.

1.2. Kohdeyleisö

Tämän dokumentin kohdeyleisö on Yritys-hankkeen ohjausryhmä, määrittelijät, projektipäälliköt, toteuttajat ja testaajat.

1.3. Termit

Termi	Selitys
Sharepoint	Microsoftin selainpohjainen tuote, joka tarjoaa välineet asiakirjojen hallintaan ja selainpohjaisen tiedonhallintasivuston rakentamiseen.
MS Exchange	Microsoftin sähköpostipalvelin (http://en.wikipedia.org/wiki/Microsoft_Exchange_Server)
WSS	Microsoft Windows Server 2003 ja 2008-versioissa oleva ilmainen lisäosa joka tarjoaa verkkoportaalin ominaisuuksia. (http://en.wikipedia.org/wiki/Windows_SharePoint_Services)
MOSS	Microsoft Office Sharepoint Server on WSS-toiminnallisuuksia laajentava kaupallinen tuote. (http://en.wikipedia.org/wiki/Microsoft_Office_SharePoint_Server)
WebPart	Toiminteen tarjoava yksittäinen verkkosivun osa. (http://en.wikipedia.org/wiki/Web_part)

Alma Stankovic

26.5.2010

1.4. Versionhallinta

Kirjoittaja	Versio	Muutokset	Tarkastaja
SNa	0.1	Pohja	
ASt	0.2	Testausraportin pohja	
ASt	1.0	Yritys-hankkeen testaus Visualwebin testiympäristössä (172.16.**.**)	

Alma Stankovic

26.5.2010

2. Tarkoitus ja kattavuus

Tässä dokumentissä kuvataan testauksen etenemistä. Dokumentissä kuvataan sitä, miten eri testit menevät läpi. Testauksessa testataan vain toiminnallisuudet, eikä testauksessa oteta huomioon itsepalveluportaalin ulkoasua.

Testaus suoritetaan Visualwebin toimitiloissa (Helsingin/Vaasan toimisto). Testausympäristö sijaitsee WSS-palvelimella, jonka IP-osoite on 109.69.**.***. Kohdepalvelimella testaus suoritetaan Internet Explorer 8 -selaimella.

Alma Stankovic

26.5.2010

3. Testausraportti

Nro	Testattava kohde	VW kommentit	OK/EI	Yritys kommentit	OK/EI
1	**Kontaktit**				
1.1	**Listaus-oletusnäkymä**				
	Oletusnäkymä linkki		OK		
	Luo uusi kontakti linkki		OK		
	Listaus näyttönimi	Oletusarvoisesti listaus on järjestetty aakkosjärjestyksessä kontaktin näyttönimen mukaan.	OK		
	Listaus organisaatio		OK		
	Listaus sähköposti	*ASt 18.5: HUOM: listaa domainin mukaan!*	OK		
	Haku: organisaatio ja/tai sähköposti ja/tai näyttönimi	Organisaatio: Wildcard* Sähköposti: *Wildcard* Näyttönimi: Wildcard*	OK		

Alma Stankovic

26.5.2010

Nro	Testattava kohde	VW kommentit	OK/EI	Yritys kommentit	OK/EI
1.2	**Luominen**				
	Määrittelyn mukaiset kentät		OK		
	Pakolliset kentät	Näyttönimi & sähköpostiosoite	OK		
	Tabulaattori		OK		
	Luo kontakti-painike	*ASt 18.5: Lukee lähetä*	EI		
	Tyhjennä kentät-painike	*ASt 18.5: Painike puuttuu*	EI		
	Tietojen talletus		OK		
	Ilmoitus käyttäjän lisäyksestä		OK		
	Ilmoitus, jos sähköpostiosoite jo löytyy		OK		
	Oletusnäkymä linkki	Kun lomake on tyhjä, ei palaa oletusnäkymään.	EI		
	Käyttäjätunnus	Generoidaan kontaktille määrittelyn kohdan 3.2.4 mukaan.	OK		

Alma Stankovic

26.5.2010

Nro	Testattava kohde	VW kommentit	OK/EI	Yritys kommentit	OK/EI
1.3	**Muokkaaminen**				
	Lomakkeen otsikko	” Lomakkeella on otsikko ”Muokkaa kontaktia”.”	EI		
	Määrittelyn mukaiset kentät		OK		
	Pakolliset kentät	Näyttönimi & sähköpostiosoite	OK		
	Käyttäjätunnus	Ei muokattavissa	OK		
	Tabulaattori		OK		
	Tallenna-painike	*ASt 18.5: Lukee lähetä*	EI		
	Poista-painike	*ASt 19.5: Ei varmista poistoa. Ei poista listauksesta, kontaktin voi löytää jakelulistojen jäsenhausta.* Määrittely 3.2.3.1: Poistaa kontaktin tietokannasta eli merkitsee statuksen (Kontakti.KN_Status) nollaksi ja mahdollisista jakelulistan jäsen taulusta (Ryhma_jasen) merkitsee käyttäjän statuksen (RJ_Status) nollaksi. Varmistaa käyttäjältä poiston ennen poistoa.	EI		
	Peruuta-painike	*ASt 18.5: Painike puuttuu* Määrittely 3.2.3.1: Palaa edelliseen näkymään	EI		

Alma Stankovic

26.5.2010

Nro	Testattava kohde	VW kommentit	OK/EI	Yritys kommentit	OK/EI
	Jakelulistojen listaus ja linkki	*ASt 18.5: Listaus puuttuu, esim. Beeta-käyttäjä kuuluu jakelulistaan.* Määrittely 3.2.3.1: Lomakkeella listataan jakelulistat joiden jäsenenä kontakti on. Rivi toimii linkkinä jakelulistanmuokkauslomakkeelle, lomake avataan uuteen ikkunaan.	EI		
	Tietojen talletus		OK		
	Ilmoitus tietojen tallentumisesta		OK		
	Ilmoitus, jos sähköpostiosoite jo löytyy		OK		
	Oletusnäkymä linkki	Kun lomake on tyhjä, ei palaa oletusnäkymään.	EI		
			OK		

Testipvm:	23.4.2007	Testaaja: Alma Stankovic, Visualweb Oy			
Testi:	1				
Testin kohde:	Käyttäjän luonti - ei ääkkösiä				
Testin tarkoitus:	Käyttäjän luonnin toimivuus				
Testin kulku:	Toimenpiteet:	Kyllä/Ei	Toiminto OK	Toimivuus OK	Lisätiedot:
1	Uusi käyttäjä - ei ääkkösiä tai erikoismerkkejä (visualweb testaa)		x	x	Käyttäjätunnus = käyttäjälle annettu nimi. Käyttäjänimeä ei voi muokata.
2	Salasanan asetus - (visualwebtestaa)		x	x	
3	Profiilin asetus - (kaikkia koitettu asettaa, CustomerAdmin valittu)		x		CustomerAdmin ainut joka toimii Kts. Virheraportti - Virhe 1
4	Oletussivun määritys - (460923)		x	x	
5	Tietokanta-asetus (Kyllä, Tapahtuma- ja Koulutuskalenteri)		x	x	
6	Käyttäjätunnus välittyy titleen			x	
7	Käyttäjän poistaminen		x	x	Käyttäjän poistamisen jälkeen se poistuu myöskin ryhmästä.

Testipvm:	23.4.2007	Testaaja: Alma Stankovic, Visualweb Oy			
Testi:	2				
Testin kohde:	Käyttäjän luonti - ääkkösillä				
Testin tarkoitus:	Käyttäjän luonnin toimivuus				
Testin kulku:	Toimenpiteet:	Kyllä/Ei	Toiminto OK	Toimivuus OK	Lisätiedot:
1	Uusi käyttäjä - (älämölötä)		x	x	
2	Salasanan asetus - (älämölötä)		x	x	
3	Profiilin asetus - (kaikkia koitettu asettaa, Content Manager valittu)		x	x	CustomerAdmin ainut joka toimii Kts. Virheraportti - Virhe 1
4	Oletussivun määritys - (460923)		x	x	
5	Tietokanta-asetus (Kyllä, Tapahtuma- ja Koulutuskalenteri)		x	x	
6	Käyttäjän nimi välittyy titleen			x	
7	Käyttäjän poistaminen				Käyttäjän poistamisen jälkeen se poistuu myöskin ryhmästä.

Testipvm:	23.4.2007		
Virhe 1			
Testi:	Alkutilanne:	Virhe:	
1 ja 2	Tunnuksella jokin muu profiili kuin CustomerAdmin	Sivuston rakenteessa rakennepuu ei avaudu.	
		Sivuston rakenteessa ei näy Vaconin alaprojekteja.	
Virhe 2			
Testi:	Alkutilanne:	Toimenpiteet:	Lopputulos:
5 ja 6	Käyttäjä luodaan Admin-tunnuksella CustomerAdmin profiililla. Lisätään käyttäjä Admintesti-ryhmään. Käyttäjän asetuksia muutetaan Admin-tunnuksella. Ja käytetään toista Remotedesktop-sessiota tunnuksen testaukseen.	1. Aktiivinen-toiminto: Kyllä Siteadmin-toiminto: Ei	Käyttäjältä estetään pääsy Siteadmin-järjestelmään
		2. Poistutaan käyttäjänhallinnasta	
		3. Yritetään kirjautua ko. Tunnuksella toisella Remotedesktop-ikkunassa.	Ei käyttöoikeuksia Code(13)
		4. Aktiivinen-toiminto: Ei Siteadmin toiminto Kyllä	Käyttäjälle annetaan oikeus päästä Siteadmin-järjestelmään, samalla poistetaan käyttäjä aktiivista.
		5. Poistutaan käyttäjänhallinnasta	
		6. Yritetään kirjautua ko. Tunnuksella	Käyttäjä pääsee järjestelmään vaikka poistettu aktiivista.